Mobile graphical user interface and controller of fast pulse generator designed in LabVIEW

Primož Mekuč, Matej Reberšek

Fakulteta za elektrotehniko, Univerza v Ljubljani
E-mail: primoz.mekuc@student.uni-lj.si, matej.rebersek@fe.uni-lj.si

Abstract

We developed user friendly mobile graphical user interface and controller of fast bipolar pulse generator. Graphical user interface was made on Nexus 7 with Data Dashboard for LabVIEW and controller was made with LabVIEW 2013 on MyRIO. The whole system was developed quite quickly, is user friendly and inexpensive. It was made with only one programming language on relatively inexpensive equipment. The user of our device sets seven electrical parameters of high-voltage bipolar pulses, arms the device and then starts the pulse delivery on tablet. We demonstrated that using LabVIEW developed controller on MyRIO pulses with resolution of 25 ns can be generated.

1 Introduction

Modern devices are often equipped with advanced graphical user interfaces (GUI). This is mainly because they are in general sophisticated, designed to be as much as possible ergonomic and hardware for GUI is nowadays relatively inexpensive. Most of modern GUIs are running on Linux or embedded Windows and recently also on Android. GUIs are usually designed in development environment for example Visual Studio or LabVIEW and programmed in one of visual programming language.

In our laboratory, we are developing biomedical devices mostly for electroporation [1], [2] [3]. In order to control such devices we have recently developed GUI on PC with touchscreen and controller on field-programmable gate array (FPGA) [4]. GUI was designed in Visual Studio and runs on Windows CE, and controller was designed with hardware description language (VHDL). GUI and controller are interconnected with USB 2.0. For such connection between the devices we wrote USB driver for Windows CE and firmware for USB 2.0 microcontroller in C programming language [5]. Such design of GUI and controller meets the imposed medical safety standards [6]. However, such system is difficult and relatively expensive to develop. Consequently we are looking for less demanding and less expensive way of developing a graphical user interface and controller for our experimental devices.

We are currently developing generator of fast high-voltage bipolar pulses that will be used in biomedical research. To control such generator the operator needs to set amplitude of the pulses, pulse duration, delay between positive and negative pulse, number of bipolar pulses in one burst, delay between the bipolar pulses, number of bursts and delay between bursts. To enter so many electrical parameters into device we need user friendly GUI and to generate control signals with nanosecond resolution we need fast controller.

Our aim in this study was to design GUI and controller of fast pulse generator, which is easy to develop, user friendly and inexpensive. For this purpose we had to choose appropriate visual programming language, which allows quick and easy development of user friendly GUI and software for the controller. We also had to choose appropriate hardware platform, which is inexpensive, has user friendly interface and has full support in the selected visual programming language. Moreover, both, visual programming language and hardware platform, had to be able to control pulse generator in nanosecond intervals.

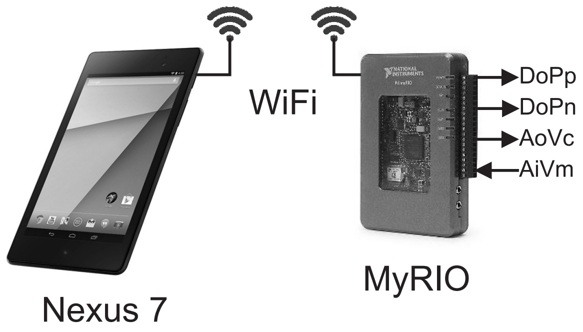


Figure 1: Block scheme of the developed mobile GUI and controller of fast pulse generator. Mobile GUI is developed on Nexus 7 and controller of fast pulse generator is developed on MyRIO. Both, Nexus 7 and MyRIO, are interconnected over WiFi. Controller of fast pulse generator has two digital outputs to control positive pulse (DoPp) and to control negative pulse (DoPn), one analog output to control voltage of the power supply of the generator (AoVc), and one analog input to monitor voltage from the power supply (AiVm).

2 Material and methods

LabVIEW 2013 (National Instruments, USA) was used as a system-design platform for the development of our GUI and controller of fast pulse generator. Google tablet

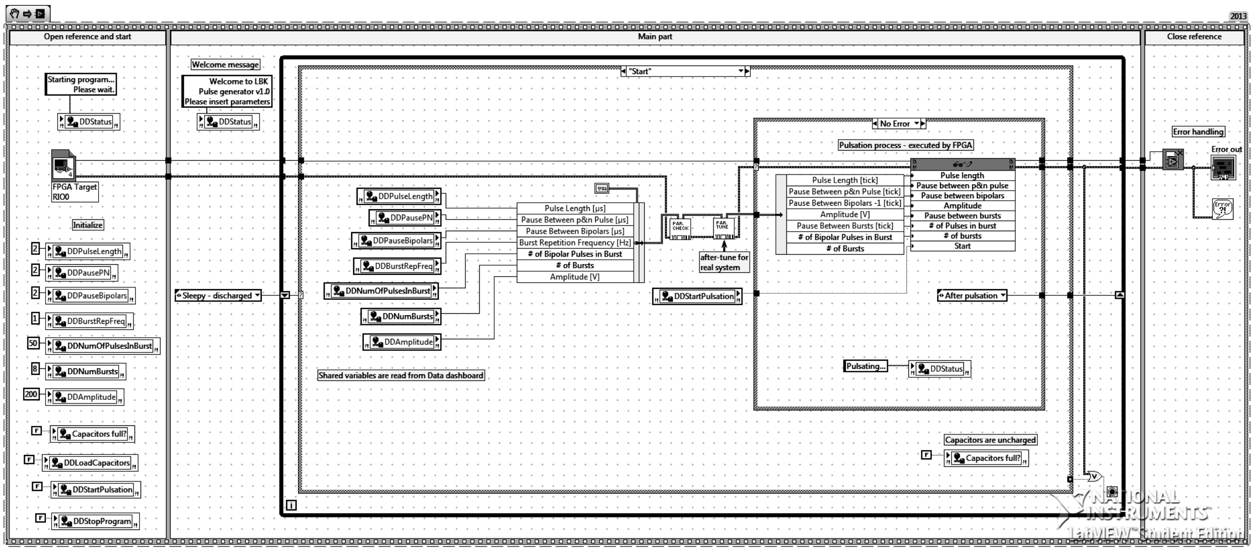


Figure 2: Labview code of real-time (RT) part which runs on ARM Cortex-A9 in MyRIO. RT part communicates with GUI designed with Data Dashboard and controls FPGA in MyRIO. There are three main sequences in our RT code made by time-sequence structure i.e. initialisation, main part, and closing reference and error handling. In the main part there is a basic state machine which handles with Data Dashboard shared variables and controls the FPGA part.

Nexus 7 (ASUS, Taiwan) was used as a hardware platform for GUI, and MyRIO (National Instruments, USA) was used as a hardware platform for the controller (Figure 1). We also used Data Dashboard for LabVIEW (National Instruments, USA) to design GUI on Nexus 7.

2.1 Hardware platform

Nexus 7 is a medium size (198.5x120x10.5 mm) tablet with WiFi, and 7.0 inches and 800x1280 pixels display. Nexus 7 runs the native Google Android operating system. MyRIO is a portable reconfigurable I/O device with 10 analog inputs, 6 analog outputs and 24 digital I/Os. Main core of MyRIO is Zynq 7010 (Xlinx, USA) with Dual ARM Cortex-A9 and field-programmable gate array (FPGA). MyRIO offers 40 MHz onboard clock for the FPGA and WiFi connectivity.

Bipolar pulse generator that we are going to control with our GUI and controller is a high-voltage and high-frequency H-bridge digital amplifier with high-voltage power supply. Amplifier has two digital inputs to control positive and negative pulse, and power supply has one analog input to control output voltage and one analog output to monitor output voltage.

We have set MyRIO as a wireless network host and interconnected it with Nexus 7 (Figure 1). We have used two digital outputs of MyRIO, one to control positive pulse (DoPp) and other to control negative pulse (DoPn). We have also used one analog output of MyRIO to control voltage of the power supply of the generator (AoVc), and one analog input to monitor voltage from the power supply (AiVm).

2.2 Software platform

In LabVIEW 2013 with MyRIO drivers we have written all the logic of the device. Firstly, we designed algo-

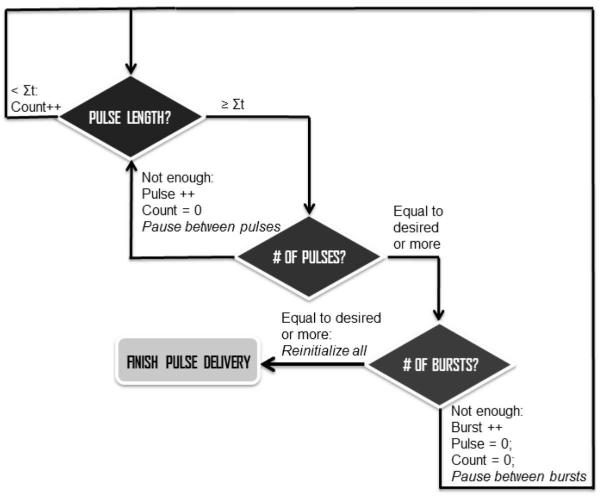


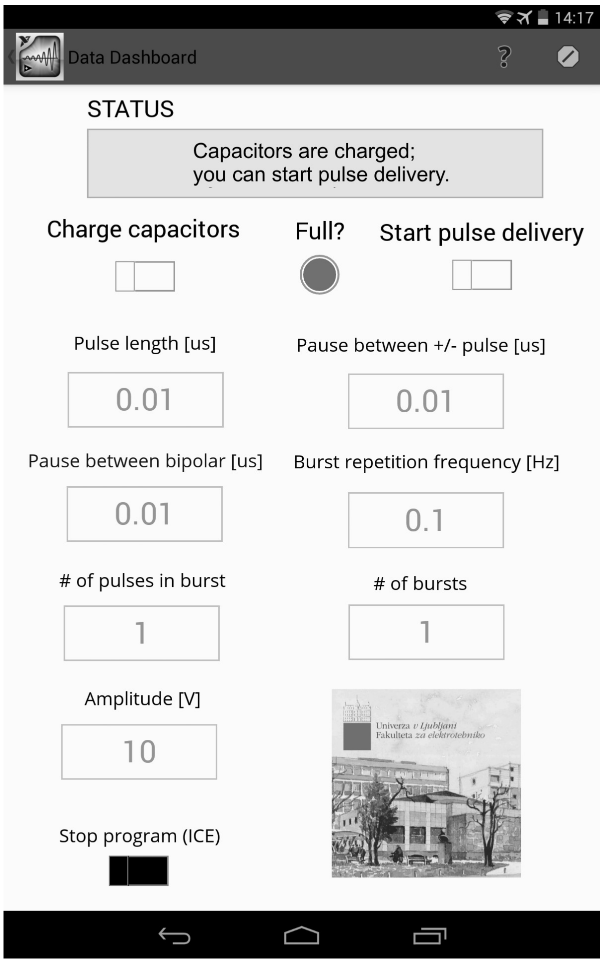
Figure 3: Basic flowchart of the FPGA program which controls the pulse delivery sequence. By counting the ticks of the onboard clock, the FPGA program checks the length of the pulses the number of pulses in one burst and number of bursts. After each condition is completed pause between the pulses or the bursts is generated.

rithm (Figure 3) for FPGA with LabVIEW FPGA module. FPGA in our device controls all the I/Os of the pulse generator. Time chronology is assured by time-sequence structure state machine. Each state is representing pulse or pause. In states time-length condition is checked and appropriate outputs are set to true/false. Afterwards, we programmed real-time (RT) part with LabVIEW RT module. RT part runs on ARM Cortex-A9, and communicates with Data Dashboard and controls FPGA. First sequence is initialisation, then main part and at last closing reference and error handling. We have used basic state machine which is running until stop button is pressed or error is present; states are: Sleepy-discharged, Charge capacitors, Sleepy-charged, Start, After pulsation and Shutdown. Shared variables connect data with tablet over WiFi, which is established by MyRIO. (Newest patch for MyRIO enables that is wireless network host and other devices connect to it - therefore generator is not dependent on WiFi accessibility)

3 Results

Program in LabVIEW is timely divided by time-sequence structure into three main parts (Figure 2).

First sequence initialises FPGA reference and all shared variables shown in Data dashboard GUI. Last sequence closes the reference and carries out error handling.


14:17
Data Dashboard

STATUS

Capacitors are charged;
you can start pulse delivery.

Charge capacitors | Full? | Start pulse delivery

Pulse length [us]: 0.01
Pause between +/- pulse [us]: 0.01
Pause between bipolar [us]: 0.01
Burst repetition frequency [Hz]: 0.1
# of pulses in burst: 1
# of bursts: 1
Amplitude [V]: 10
Stop program (ICE)

Figure 4: GUI with all the buttons and controls through which the user can enter seven electrical parameters of the pulses and starts or stops the pulse delivery.

Middle sequence is core of program and its main structure is basic state machine which steers between states. First state "Sleepy-discharged" initialises shift register and is the first state. If the user wants to start delivering pulses, program inform that capacitors should be charged first. On the other hand when user presses charge capacitors button program continues to "Charge capacitors". In this state analog output for linear power supply outputs appropriate output voltage for capacitor charging and LED "Full?" lights, so the user knows that capacitors are full. Right after program automatically goes to "Sleepy-charged". In this state there are two options as well - if user tries to press charge again, program ignores request and outputs message in status string (Figure 4). Otherwise when user is ready to apply pulses, start button immediately starts process. Parameters are read from shared variables, packed into cluster and modified in two subVIs. Afterwards they are passed to FPGA program by IO reference. In FPGA program 1-tick i.e. 25ns (reciprocal of 40 MHz clock) accurate timing for digital outputs are assigned with respect to to parameters received from Data dashboard. Basic workflow is described in Figure 3. When pulsation completes, program goes further to "After pulse delivery" state where confirmation message is shown if everything went right and LED indicator indicating charge is turned off. Afterwards it returns back to "Sleepy-discharged". Additional high priority state is "Shutdown". As soon as user press "Stop program [ICE]" or error is perceived by LabVIEW, program goes to shutdown state and turns off.

GUI (Figure 4) is built in National Instruments Data Dashboard programme for Android. It enables communication over shared variables with any device running LabVIEW. We made dashboards is controlling the pulsation. We tried to build as simple and intuitive interface as possible, so even non-technical people could use it. In the bottom right corner there is a picture symbol of our Faculty. When user presses start button in top right corner interface begins to work.

By developed controller we can generate control pulses with time resolution of 25 ns (Figure 5) and accurately generate control pulses with respect to entered electrical parameters (Figure 6).

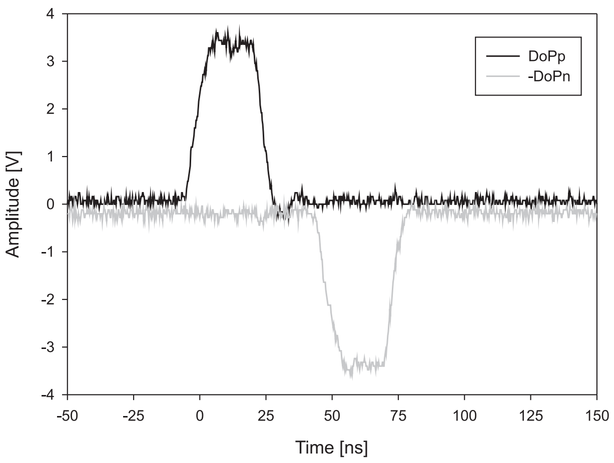


Figure 5: The time resolution of digital outputs. The quant and the minimum length of our pulses and pauses is 25 ns. DoPp is a digital output to control positive pulse and DoPn is a digital output to control negative pulse.

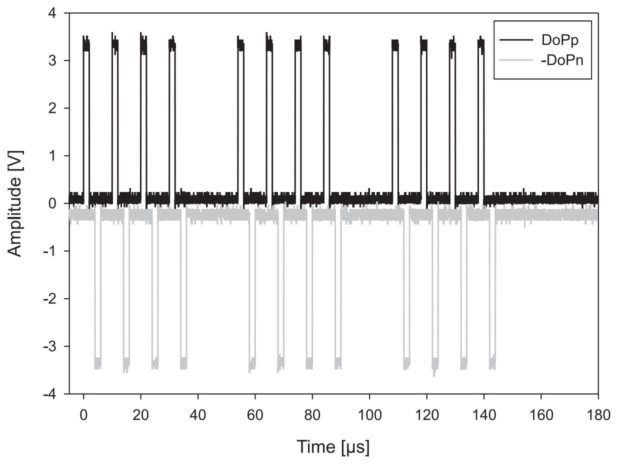


Figure 6: Digital outputs at generating pulses with 2 us length, 2 us pause between positive and negative pulse, 4 us pause between bipolar pulses, 20 kHz burst frequency, 4 pulses in one burst and 3 bursts. DoPp is a digital output to control positive pulse and DoPn is a digital output to control negative pulse.

4 Discussion and conclusions

Many different hardware platforms from industrial to consumer have been considered to be used for our GUI and controller. We find industrial hardware platforms for GUI much more expensive than commercial. However, commercial platforms are more or less mobile with mostly wireless connections. On the other hand, this might also be an advantage as such platform could be used to control more than one device. We have decided to use Google's tablet Nexus 7 (ASUS, Taiwan) as a hardware platform for our GUI, since it runs native Android and 7" size is most appropriate to be hold in one hand by experiment performer. Tablet has very sensitive touchscreen as well which enables accurate and gentle control of the device even with laboratory gloves on.

We chose LabVIEW (National Instruments) as a system-design platform because it enables us to design GUI and controller with only one development environment and because it supports many different operating systems (Android, iOS, Windows mobile) and hardware platforms. It is very convenient to design such control applications and MyRIO gave us time accuracy with FPGA. We decided for basic state machine, since event-driven state machine cannot be triggered by shared variables using basic modules that student edition of LabVIEW includes (DSC module is required). At first we have tried to design whole program in real-time, but soon we realized that real-time (RT) programme is time consuming and does not allow to run such time-precise application. So we decided to program FPGA included in MyRIO (see Hardware platform). Decision caused us few smaller problems as well, since there is only stunted set of function blocks that can be executed and program had compiled on average for 20 minutes (whereas RT compiles only few seconds).

We had a chalange to connect Data dashboard and MyRIO as well. Data dashboard 2.0 is relatively a new software at the time of writing this article. At first we had problems connecting shared variables especially when MyRIO was a host. There are a lot of IPs - from tablet, RIO, RIO's network etc. After reading some white papers and forums and discussion with support service we found a working solution. Otherwise establishing communication is convenient, does not require special knowledge and is not time consuming.

Another advantage of our approach is reusability and modularity of the parts of the system. When the researchers finish a series of experiments, we can build another analog part, design a new software and build device with totally different function. MyRIO has more than enough Analog/Digital Inputs/Outputs for connecting signals and sensors and even audio input and output. With one MyRIO being host we can even connect other MyRIOs to this same network and control more RIOs with one tablet. Therefore here are plenty of possibilities to build a modern portable device in a short time.

We demonstrated that with designed GUI and controller we can generate appropriate control signals to control our generator of fast high-voltage bipolar pulses.

5 Acknowledgments

The authors acknowledge support of their work through various grants from the Slovenian Research Agency. Research was conducted in the scope of the EBAM European Associated Laboratory (LEA).

References

[1] T. Kotnik et al. "Cell membrane electroporation- Part 1: The phenomenon". In: *IEEE Electrical Insulation Magazine* 28.5 (2012), pp. 14–23.

[2] S. Haberl et al. "Cell Membrane Electroporation – Part 2: The Applications". In: *IEEE Electrical Insulation Magazine* 29.1 (Feb. 2013), pp. 19–27.

[3] M. Reberšek et al. "Cell membrane electroporation-Part 3: the equipment". In: *Electrical Insulation Magazine, IEEE* 30.3 (2014), pp. 8–18.

[4] D. Pavliha, M. Reberšek, and D. Miklavčič. "A graphical user-interface controller for the biomedical high-voltage signal generator". In: *Journal of Electrical Engineering and Computer Science* 78 (2011), pp. 79–84.

[5] D. Pavliha et al. "A Personal Computer as a Universal Controller for Medical-Focused Appliances". In: *IFMBE proceedings*. Vol. 16. Springer, 2007, pp. 381–384.

[6] D. Pavliha, M. Reberšek, and D. Miklavčič. "Design and Quality Assessment of the Graphical User Interface Software of a High-voltage Signal Generator". In: *Journal of Electrical Engineering and Computer Science* 78 (2011), pp. 281–286.