Gas puffing system for tokamak GOLEM

V. Fišer

May 20, 2019

Abstract

GOLEM is small tokamak. For discharges usually H or He is being used as working gas. Gas management on tokamak GOLEM consisted of slow valves. Before each discharge, valves were opened so the gas inlet and chamber evacuation was in equilibrium. No adjustment could have been done during discharge. New system described was constructed within working on my barchelor thesis at FJFI.

Currently operational system is in "work in progress state" and final system may be slightly defferent.

1 Hardware

New piezoelectric gas flow valve (Key High PEV-1) was installed paralel to existing mechanical valves. New control box was built specifically for this valve.

This control box is enclosed in standard 1U rack chasis. Components of this box may be seen at Fig. 1.

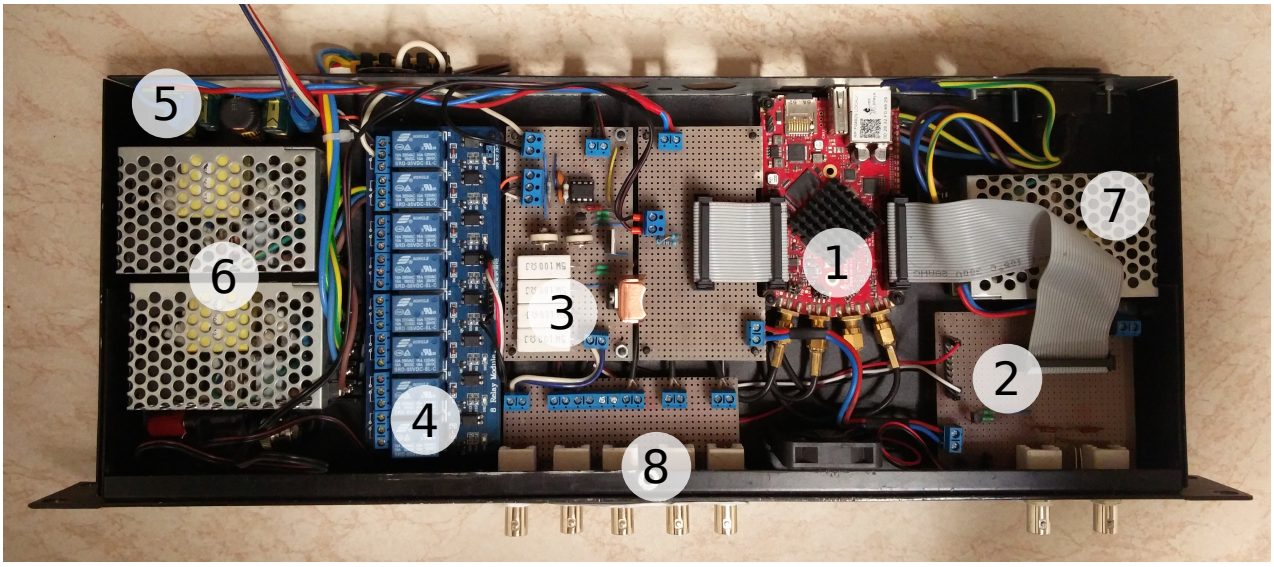

Figure 1: Parts of new control box:

1. Red Pitaya Stemlab 125-10 board
2. Trigger and digital logic
3. Amplifier
4. Relay board
5. ± 15V power supply
6. 100V (2x 50) power supplies
7. 5V power supply
8. I/O connectors (BNC coax.)

1.1 Control board

Heart of control box is Red Pitaya StemLab 125-10 board. This particular box was chosen for several reasons. It employs FPGA and provides

- 2 channels of 125 MHz 10 bit ADC
- 2 channels of 125 MHz 10 bit DAC
- 4 channels of 100 KHz 10 bit ADC
- (theoretically multiple PWM DACs)

1.2 Trigger and digital logic

GOLEM uses +5V risig edge trigger system. Digital pin of RP board (at 3.3V level) was used for trigger input. Simple divider with blue LED (operating voltage $\sim$ 3V) works fine for stabilization and protection of Red Pitaya logic.

For controlling slow digital signal, a standard relay board was incorporated. For interfacing with this board an common emitter BJT circuit was build.

Schematics of digital logic is on Fig. 2.

1.3 Amplifier

Piezo valve is controlled by 0-100V voltage signal. Since the valve is piezoelectric, it acts mostly as capacitive load. Red pitaya DAC can output -1 to +1 V. A simple electronic voltage amplifier was designed and constructed to adapt RP output to valve. Main requirements for amplifier was transition properties: high slew rate, stability and short settling time.

Schematics of the amlifier is on Fig. 3. For simplicity, no global feedback was used. This lead to shorter settling times. Drawback of this circuit is non linear behavior. This was not a big problem, since this could be simply compensated in software.

Power loss on output stage FET transistor and resistors is quite high in open state, so series of four 5W 100Ω resistors and TO220 IRF640 power MOS-FET with small heat sink was used.

Two potentiometers were used for setting input stage DC offset and gain.

Whole circuit is powered by 100 V 400 mA industrial power supply with auxiliary $\pm$15 V for operational amplifier.

2 Software

Gas puffing system needs to be contrlled from central tokamak system. In currently used mode, predefined sequence of signal ($\sim$ flow) for the valve is prepared prior to discharge, then preformed during the discharge.

2.1 Box software

Red Pitaya uses Xilinx Zynq 7010 chip, which combines FPGA and ARM CPU. The CPU runs standard linux operating system allowing us control the device easily, while FPGA handles time critical tasks. In our case, system boots ubuntu based system[1] from microSD memory card.

For FPGA, a standard image (bitsream) supplied for Red Pitaya boards "v0.94"[2] was used. One of its features is arbitrary waveform generator. This means, that predefined waveform will be "played back", optionally at an external trigger.

Control software was written in C language using an Red Pitaya C API[3]. This program reads prepared samples from text file into memory and sets the FPGA into arbitrary waveform mode and also handles auxiliary logic. Predefined waveform is loaded into the system manually using SSH prior to discharge.

System operations is controlled form central GOLEM control system using SSH. On first call (first phase of preparations, capacitor charging for tokamak) the program is run in screen shell in time of discharge preparations. In first phase of preparations (capacitor charging for tokamak) an amplifier (see 1.3) is powered on and input waveform is loaded. On second call ("arming" phase, data acquisition system activation, preparing for trigger), the program activates the arbitrary generator and sets it for external trigger. In last phase ("post discharge") the amplifier is deactivated.

2.2 Waveform preparations

The armibtrary waveform generator can take up to 16384 samples, but enables stretching it over time or/and repeating it multiple times. In our case, the longest considered time was 100ms. It meant period for one sample $\sim 6\mu$S, which was faster, than capabilities of valve, thus sufficient[4].

[1] `https://redpitaya.readthedocs.io/en/latest/developerGuide/os/debian.html`

[2] `https://rpdocs.readthedocs.io/en/latest/developerGuide/software/fpga.html`

[3] `https://github.com/RedPitaya/RedPitaya/blob/master/api/include/redpitaya/rp.h`

[4] Unlike DAC samplerate limited systems with analog filters, the signal was at the DAC level sampled much faster, so real output voltage was practically linear interpolation, not bandwith limited.

Since preparing input waveform manually was inconvenient, a simple python script was written providing possibility to superpose multiple pulses, ramps and numerical waveforms.

Source codes may be found at gitlab: `https://gitlab.fjfi.cvut.cz/fiservo3/gas-puffing-control/`.

Usage may be seen in example file `https://gitlab.fjfi.cvut.cz/fiservo3/gas-puffing-control/blob/master/waveform_generator2/generate_example.py`

File transfer is done using SSH. For linux systems, makefile with such capability is provided.

Schematics

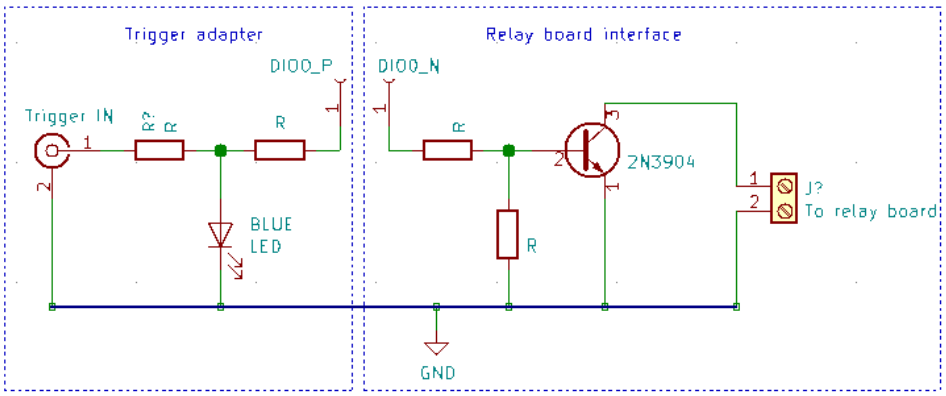

Figure 2: Trigger adapter and relay board interface

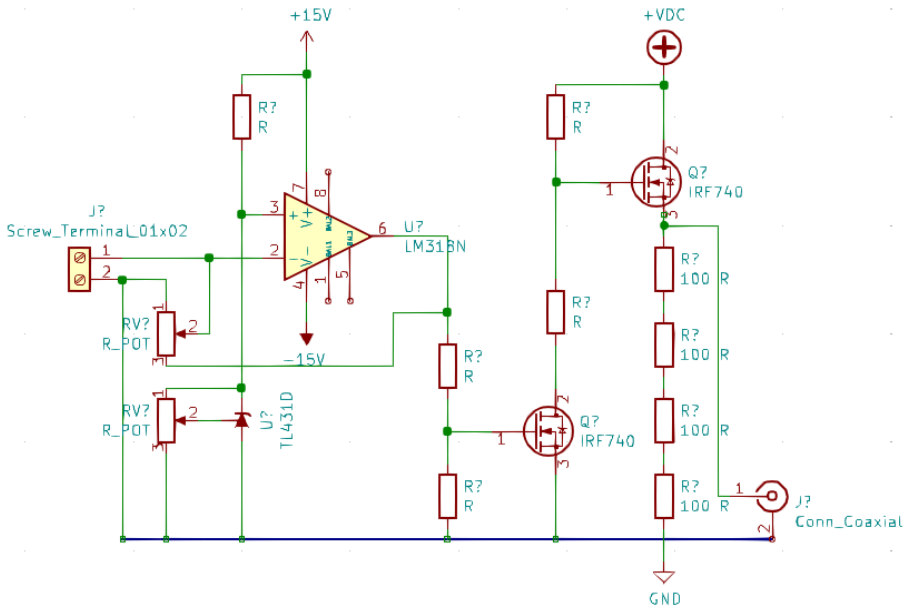

Figure 3: Shematics of amplifier