Mutual inductance based impedance simulator

EMPIR project report: LiBforSecUse, activity A1.1.4

Stanislav Mašláň, smaslan@cmi.cz, Czech Metrology Institute, 2020-02-14

Following document briefly describes experimental design of capacitive reactance simulator based on mutual inductance. The simulator is designed for calibration of battery impedance spectroscopy analysers (EIS).

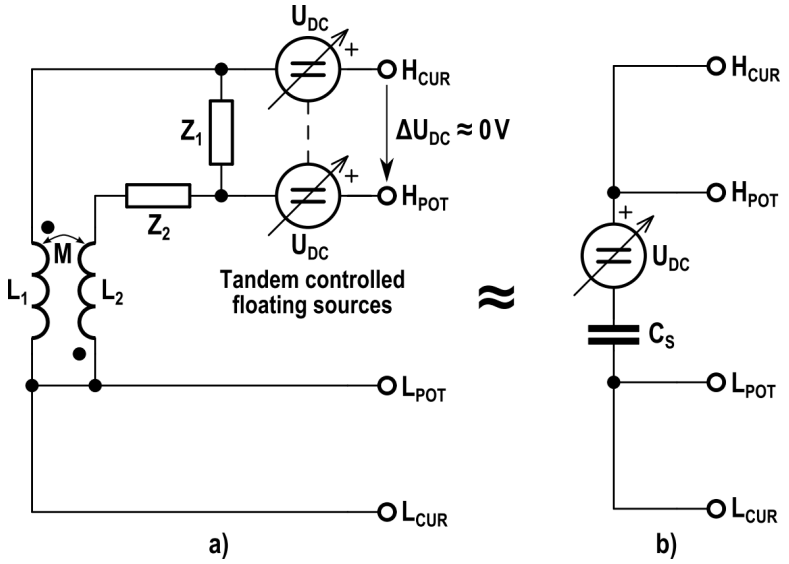

Figure 1: Principle of capacitance simulator based on mutual inductance.

The goal of the design was to achieve large capacitive reactance with simulation of DC bias voltage roughly equal to lithium cell (around 4V). The standard must be able to withstand measurement current at least 1A AC and 1A DC (2.5A peak current in both polarities).
One of the simple ways of simulating reactance is mutual inductor which can be easily designed to exhibit mutual inductances in order of nanohenry to microhenry and to be able to carry high measurement currents. If the secondary winding polarity is inverted, the apparent four terminal impedance will appear as capacitive reactance to the EIS. The basic concept is shown in Figure 1. Impedances Z_1 and Z_2 can be used to fine tune the apparent impedance, however they are not necessary.

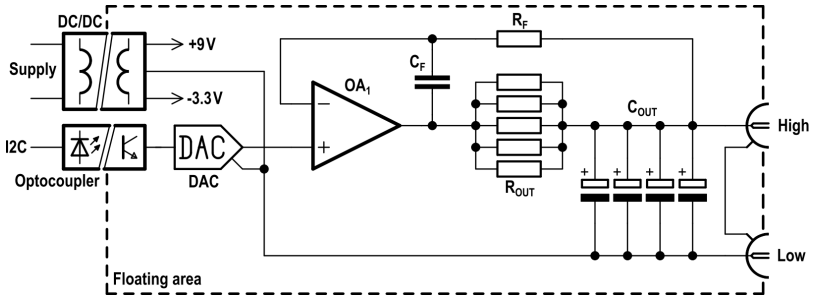

Figure 2: Principle of DC bias generator.

The challenging part of the design is however introduction of the DC bias voltage. It is not possible to simply insert high power DC source in series with mutual inductor, because its internal impedance would destroy properties of the impedance. It is also not reasonable to insert it just into the potential terminal, because then the EIS meter will see DC voltage between its Hcur and Hpot terminals. It may lead to fail indication or even damage of the EIS. So an alternative solution shown in Figure 1

was chosen. Two tandem controlled sources were used. One, high power floating DC source capable to withstand the measurement current is connected in series with the Hcur terminal. This source does not have to be very accurate. A second floating DC source is placed in series with the Hpot terminal. This source carries almost no current, but should be accurate and low noise. Both sources are set to identical voltage, so the EIS meter will see near zero voltage difference between Hcur and Hpot terminals and at the same time will see necessary DC bias of the impedance.

Design of DC bias source

Circuit design of first prototype is shown in Figure 3 or see GitHub source files [1]. Evidently the only complicated part of the design is the DC bias source. Both DC sources must be floating. For purposes of first prototype this was solved by using DC/DC converters. The current DC bias is supplied from a pair of 9V and 3V3 converters. This asymmetric supply was chosen to minimize power dissipation as the DC voltage range is 0 to 5V and the power opamp OPA548 needs at least some 3V reserve. The DC/DC converters outputs are filtered by low-ESR polymer caps. Care must be taken to not exceed maximum filter capacitance for particular converter to prevent regulator instability. The controlled DC source is based on power opamp OPA548. It is equipped by stabilizing network and the output is filtered by low-ESR capacitors. The opamp feedback is providing low output impedance for low frequencies, whereas the large output capacitance is providing low impedance for higher frequencies. The voltage of this source is generated by 12bit DAC connected via isolated I2C bus. Zero and gain are set by two trimmers. The whole source can be disabled by DC/DC converter enable inputs and in that case a pair of parallel latching relays bypasses the output of the source.

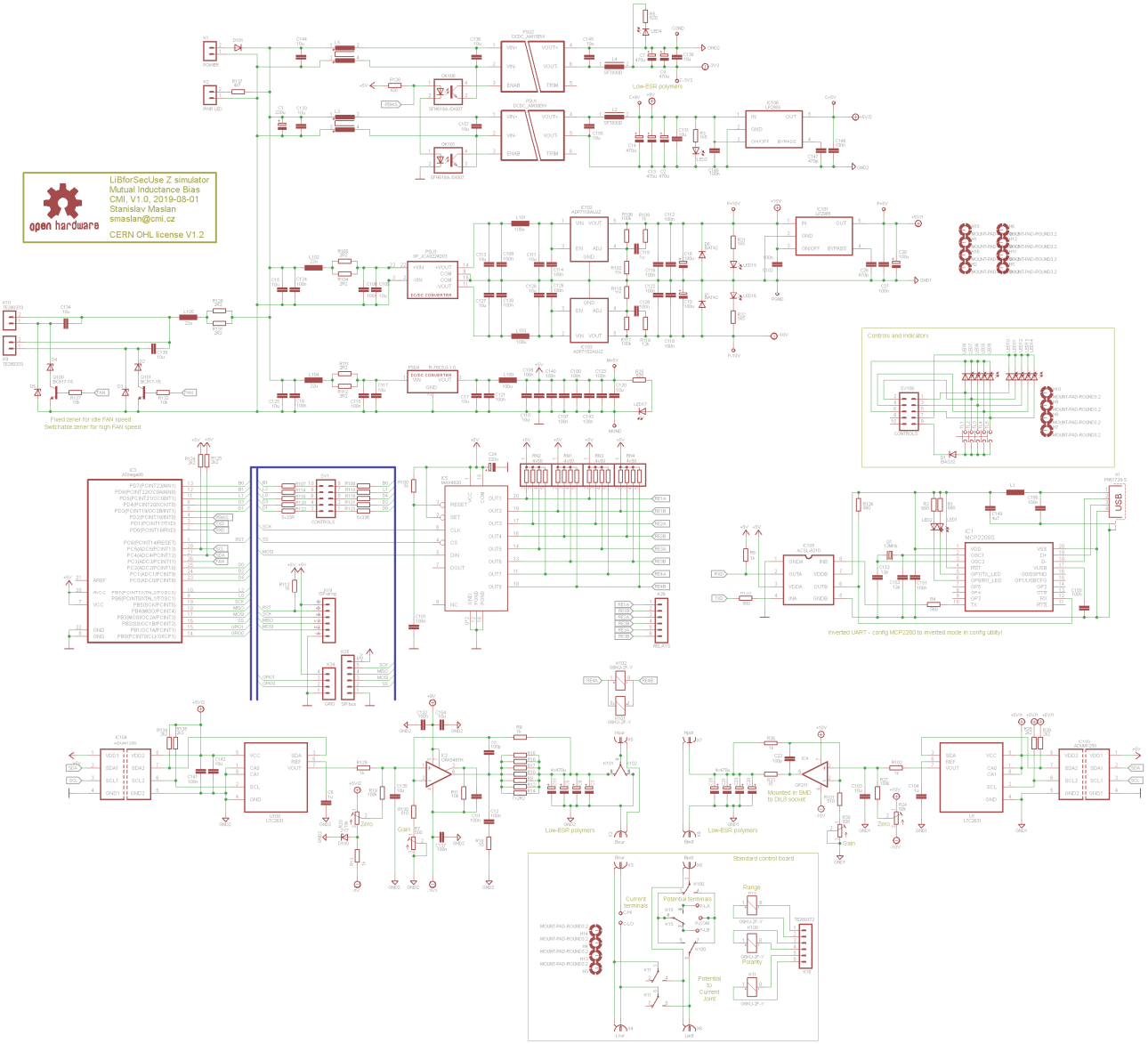

Figure 3: Circuit diagram of impedance simulator prototype.

The second source inserted to the potential terminals is of similar concept except it is supplied from low noise DC/DC converter with low noise LDO regulators and uses low power opamp OP211. Its output is also equipped by stabilizing network and the opamp and stabilizing network components must be chosen so it is stable with large output capacitance.
The standard itself (mutual inductor) is placed in separate box. It is connected via additional board with few latching relays that enables switching the polarity and range of the mutual inductor.
The whole unit is controlled by microcontroller AVR ATmega88. It is equipped by simple user interface with buttons and LED diodes and also it is connected to USB-UART converter MCP2200 for remote control. Note the MCP2200 is connected via optoisolator which inverts UART polarity, so MCP2200 configuration utility must be used to invert polarity of UART outputs.
The unit contains electronically switchable fan outputs with regulation of speed using Zener diodes. One diode is intended to be used for idle speed and the other switchable for high speed when current path DC bias is enabled.

Construction notes

Board drawings are shown in Figure 4 and Figure 5 and at GitHub [1]. The DC source unit was designed to fit in Hammond die cast box HM-1550F. The power opamp OPA548 is soldered from

bottom of the PCB, so its terminals must be bent up from the heatsink. It is screwed to the heatsink which is screwed from bottom of the PCB. The heatsink must be able to cool full power which is up to some 23W, so a fans are intended to be used.
The standard itself was placed to another box Eddyston 26908PSLA together with the relay board. Both boxes are connected via BNC connectors and Cannon 9 connector. The output terminals to the EIS are placed at the standard box. The connection between the two boxes is made using 3D printed components which are also available at GitHub [1]. Note there are no male BNCs with crimping terminals, so a flange types were used and they are connected to the coaxes via brass foil screwed to them.
The mutual inductor itself is constructed as air-core toroid so most of the field is contained. Therefore, it is not affected by near metallic objects and exhibits no significant non-linearities. The winding is made from multifilar rope of copper enamelled wires. 10 wires from the rope are connected in parallel and used as primary. The remaining 10 are connected in series to form secondary. The unit has two ranges, so one range is full 10 loops and the other range is connected to a first tap. The core of the toroid is 3D printed. See GitHub for model.

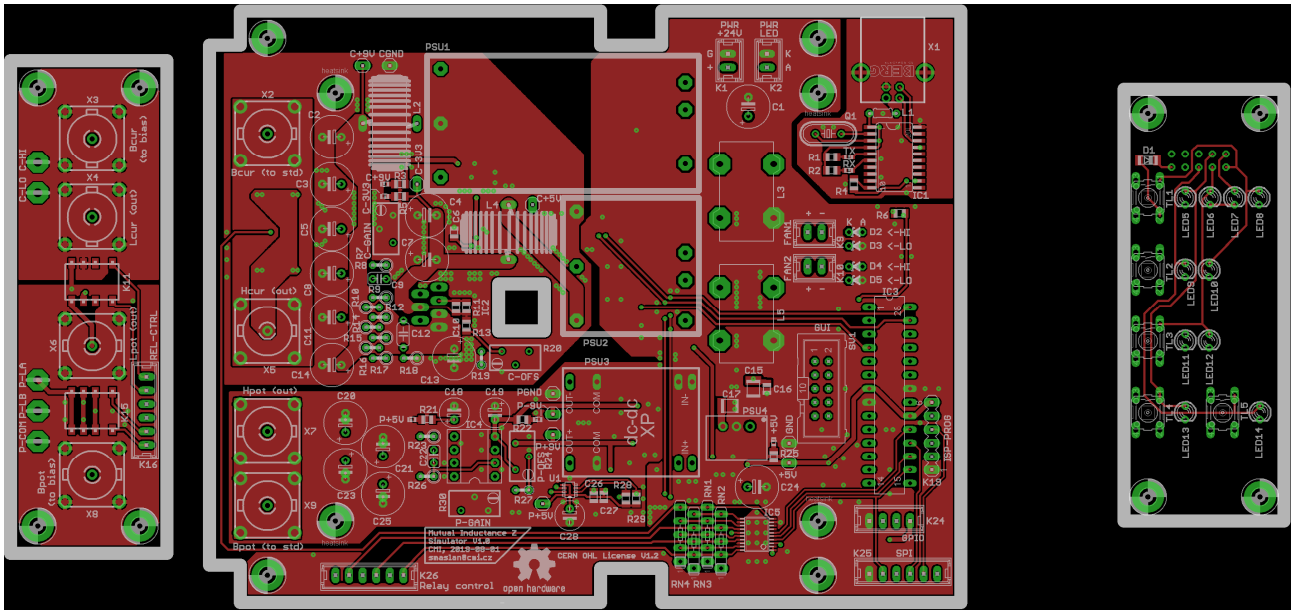

Figure 4: Top layer of PCB.

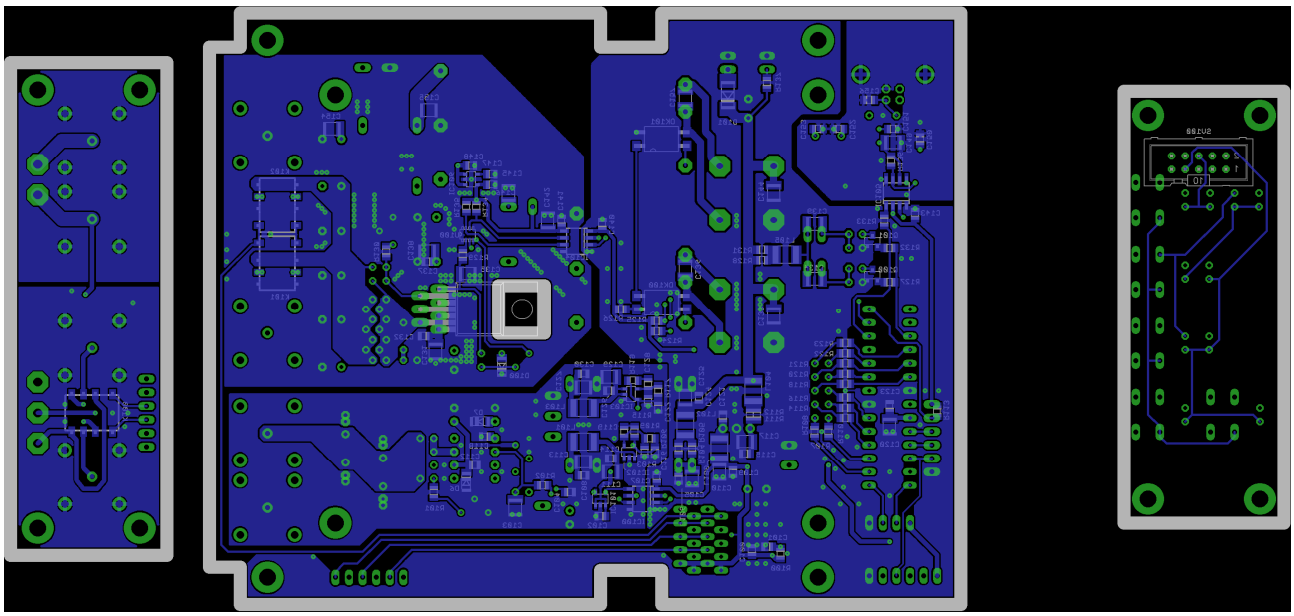

Figure 5: Bottom layer of PCB.

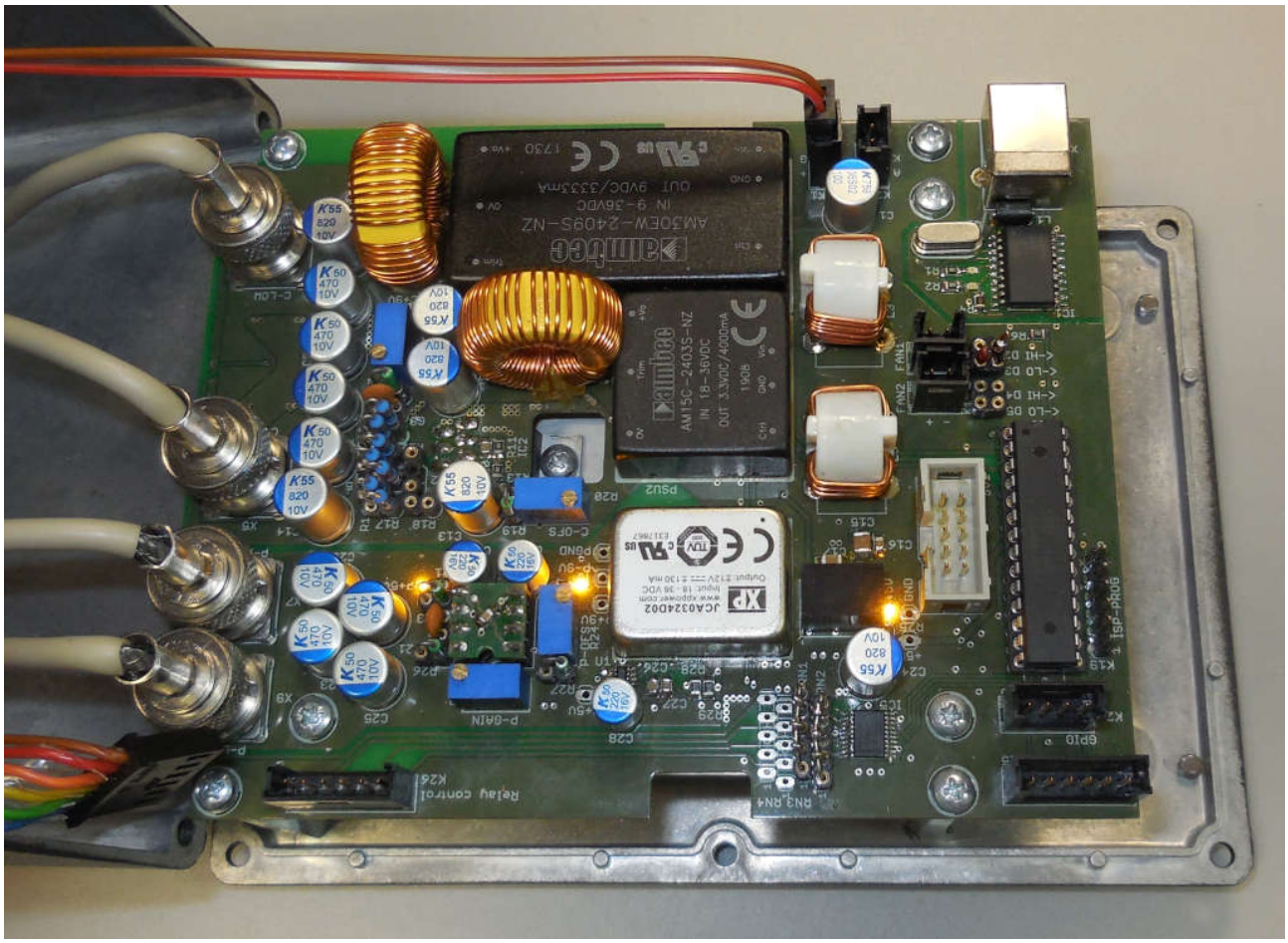

Figure 6: Dual DC bias source main PCB. Note the power opamp OPA548 mounted from bottom side of PCB to the heatsink.

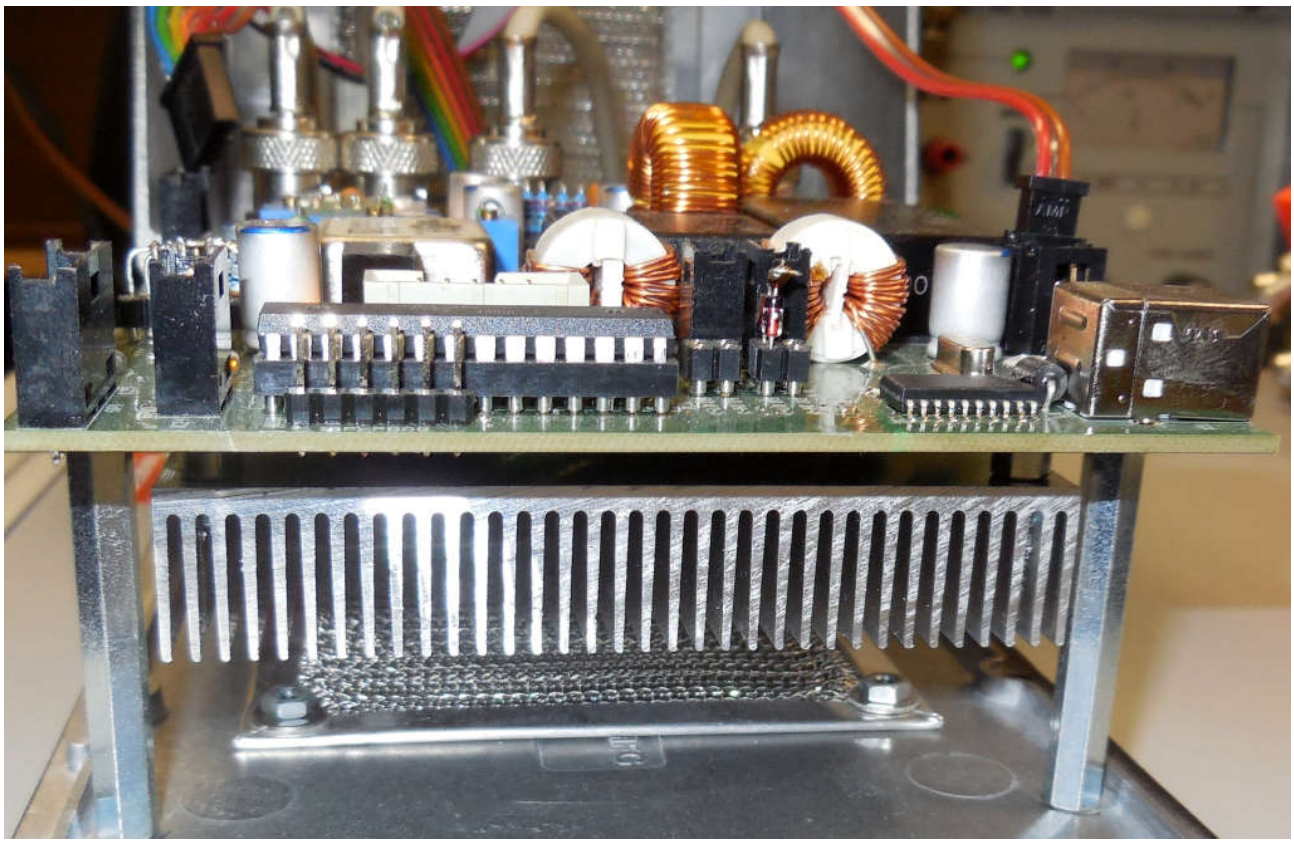

Figure 7: Heatsink mounting at bottom of main PCB.

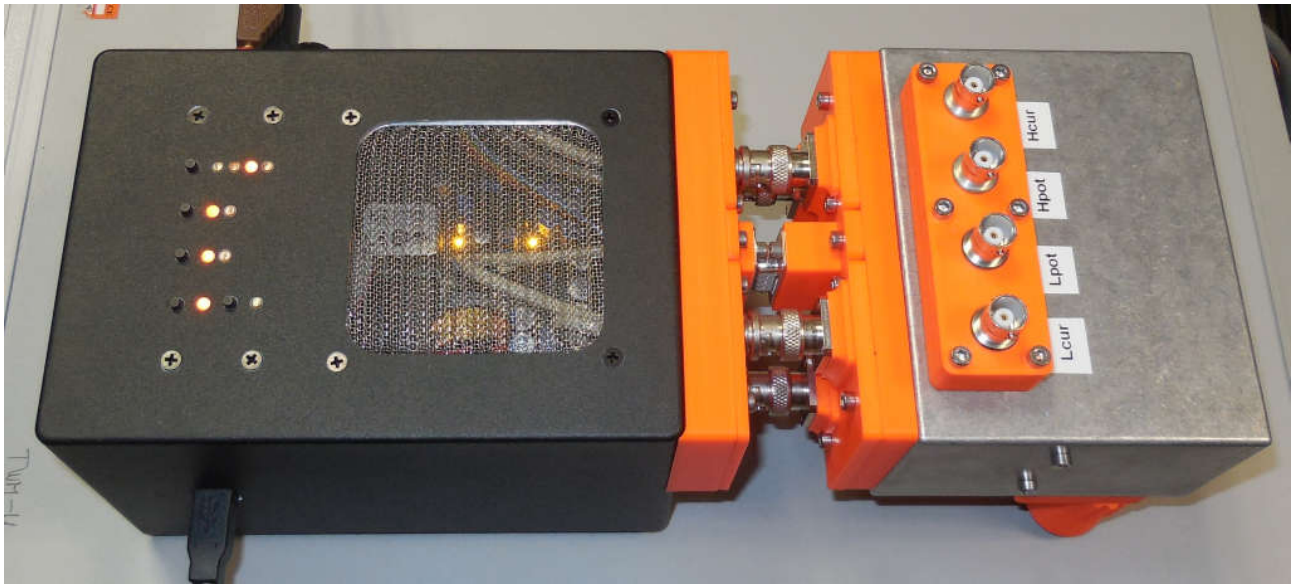

Figure 8: Complete setup. Left: DC bias sources, right: mutual inductor with terminals to EIS.

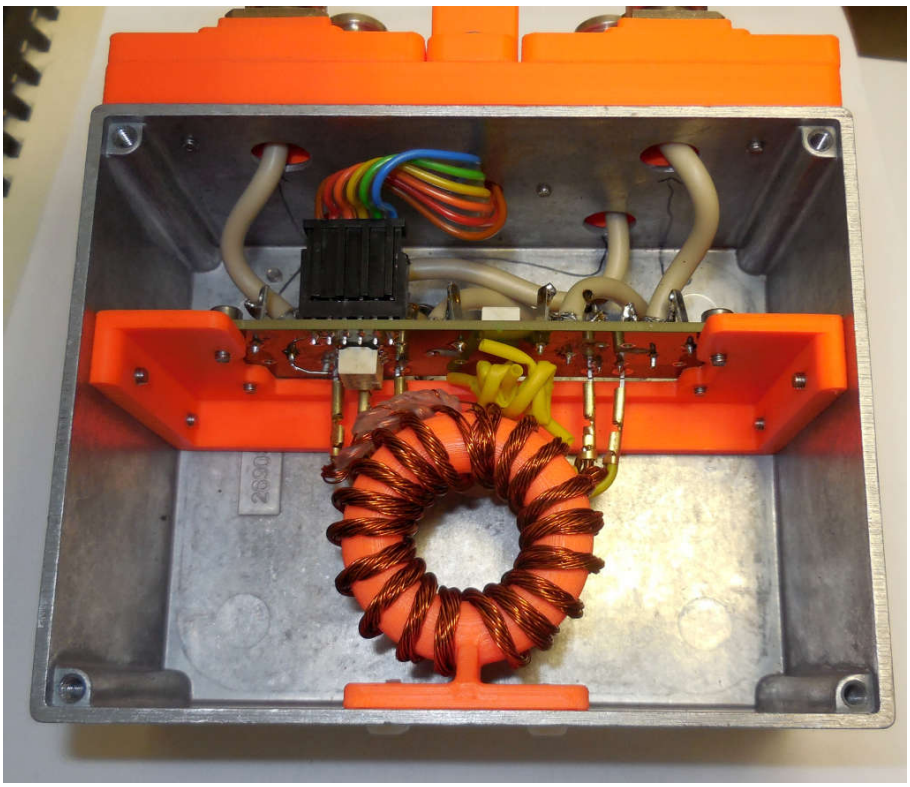

Figure 9: Separate box with mutual inductor.

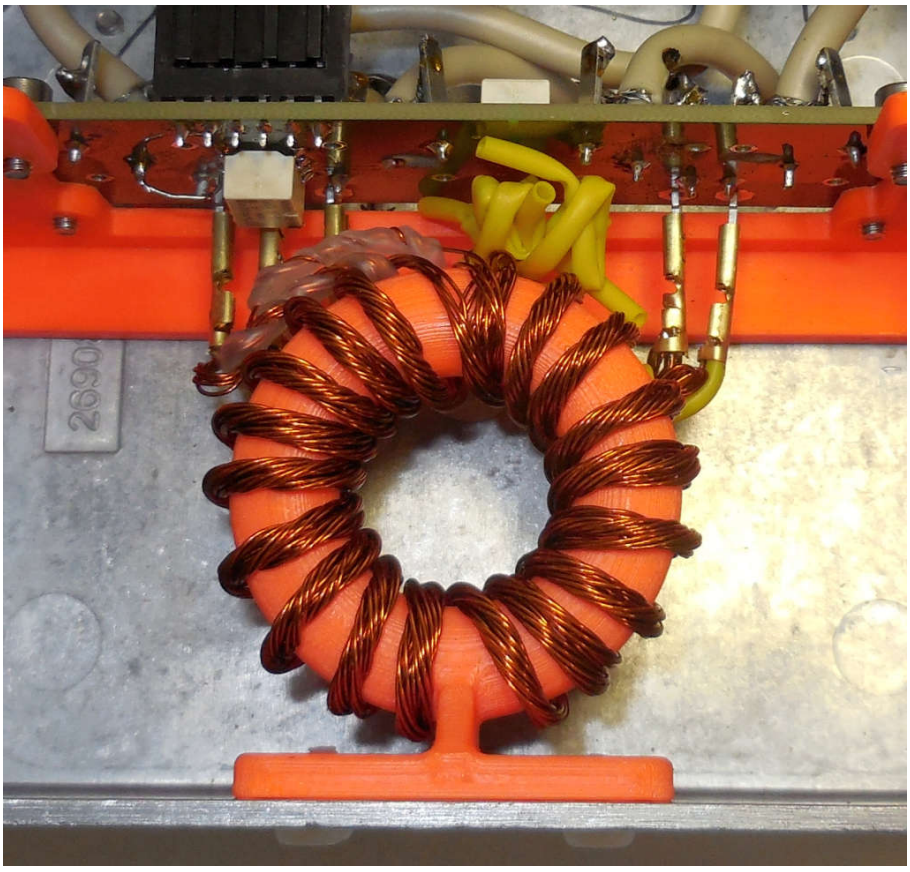

Figure 10: Detail of toroidal mutual inductor.

Firmware

The firmware for the MCU is available at GitHub. It was made in AVR Studio 4 with AVRGCC compiler. Note the unit expects operation from internal RC oscillator set to 1 MHz which is default fuse setting, so no fuses should be modified. Programming is possible only via ISP interface. It is recommended to supply the unit from regulated PSU just in case the latching relays controller is "confused" during the programming via the shared SPI interface.

Remote operation

Remote interface is set to baud rate of 9600bd, 1 stop, no parity and no flow control. It implements few basic SCPI style commands. All commands and answers are terminated by LF. Chaining by semicolon is supported. Do not exceed 127 characters in long chained commands. Errors are returned using common SYST:ERR? command.

Command	**Description**
*IDN?	Returns identification string.
*RST	Resets device.
*OPC?	Returns +1 when device is ready.
SYST:ERR?	Returns last error (buffer can take only one error).
SYST:BOOT <passcode>	Invoke bootloader. Passcode = 17IND10. This command will jump to AVR microcontroller boot section as set by fusebits. If no bootloader was programmed, it will only reset device. See text for further details.
FAN <speed>	Sets FAN speed to HI/HIGH or LO/LOW (default powerup is LO).
MODE <mode>	Sets polarity of the secondary coil to „CAP“ or „IND“.
RANGE <range>	Sets range to „1“ or „2“.
COMMON:STATE <state>	Sets state of relay tha connects current and potential terminals. Valid values are „0“, „OFF“ or „1“, „ON“.
POWER:STATE <state>	Enables or disables current DC bias source. Valid values are „0“, „OFF“ or „1“, „ON“.
BIAS:VOLT <level>	Sets both sources to identical level in millivolts. Note <level> is integer.
BIAS:POT:VOLT <level>	Sets level of potential DC bias source in millivolts. Note <level> is integer.
BIAS:PWR:VOLT <level>	Sets level of current terminal DC bias source in millivolts. Note <level> is integer.

Bootloader

Project also contains simple bootloader made in assembly code. It fits to last 512 Bytes of AVR FLASH memory. So bootsize fuse bits should be set accordingly (BOOTSZ1 = 1, BOOTSZ0 = 0). Eventually BOOTRST = 0 will make AVR jump to bootsection after powerup. For details see either bootloader ASM code or Octave bootloader m-file (both attached at GitHub [1]).

Known issues

The simulator is just a proof of concept prototype, so there are few problems that must be addressed for final version. First, the chosen DACs exhibit relatively high noise. They should be replaced by low noise types (eg AD5781) or maybe at least equipped by low noise references.
Second, the DC/DC converters produce considerable common mode noise and also have large mutual capacitance. Both problems can be addressed by active guarding. The simplest way would be to connect two DC/DC converters in series and supply the join between them from a guard buffer. That will break capacitance to supply and should also reduce the noise. At the same time the guard buffer should guard the whole DC source part of the PCB and in case of the current path source it should also guard the heatsink somehow to reduce the capacitance as much as possible.
FW should be improved to be more SCPI compliant. Especially chaining commands by semicolon should be implemented.

References

[1] GitHub, Z-sim-mutual, url: https://github.com/smaslan/Z-sim-mutual