Supplementary information to

Development and Application of a Novel Multi-Channel In Vitro Electrical Stimulator for Cellular Research

Jorge R. Cibrão [1,2], Miguel Armada[1,2], Marta F. Lima[1,2], André Vidinha-Mira[1,2], Jonas Campos[1,2], Tiffany S. Pinho[1,2], António J. Salgado[1,2], Alar Ainla[3,*], Nuna A. Silva[1,2,*]

[1] Life and Health Sciences Research Institute (ICVS), School of Medicine, University of Minho, 4710-057 Braga, Portugal

[2] ICVS/3B's Associate Lab, PT Government Associated Lab, 4806-909 Braga/Guimarães, Portugal

[3] International Iberian Nanotechnology Laboratory (INL), Braga, Portugal

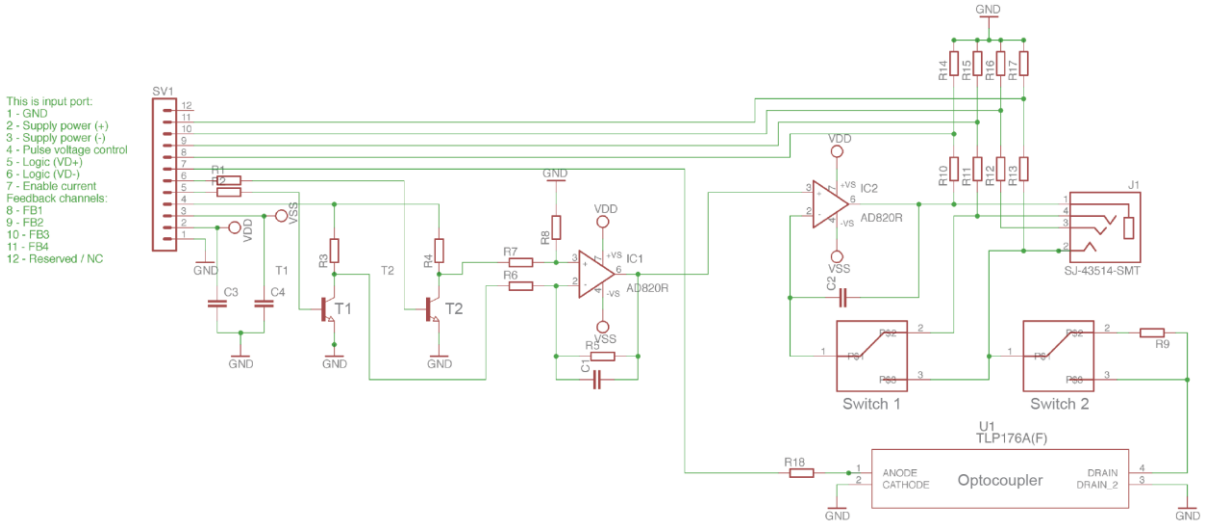
*Corresponding authors: Nuno A. Silva, nunosilva@med.uminho.pt, and Alar Ainla, alar.ainla@inl.int

Contents

1. Design of electronic controller

Electronic controller unit was designed from modular boards with individual "channel amplifiers boards" for each well. 4 amplifier boards were connected to "multiplexer boards", which were further stacked in series and connected to the "motheboard" with Arduino microcontroller. Microcontroller software was defining the eventual pulse protocols. In following we provide circuits, components and description for each of the boards. Circuitry had additional optional features, for future purposes, such as for potential monitoring and additional sensing electrodes. These features were not implemented in the presented device and were not characterized and used in this paper.

1.2. Channel amplifier board



Supplementary Figure 1. Circuit diagram of the channel amplifier board. Component values are listed in table S1. Board has two input and output connectors (SV1 and J1), which are described in table S2.

Supplementary Table 1. Components of the channel amplifier board.

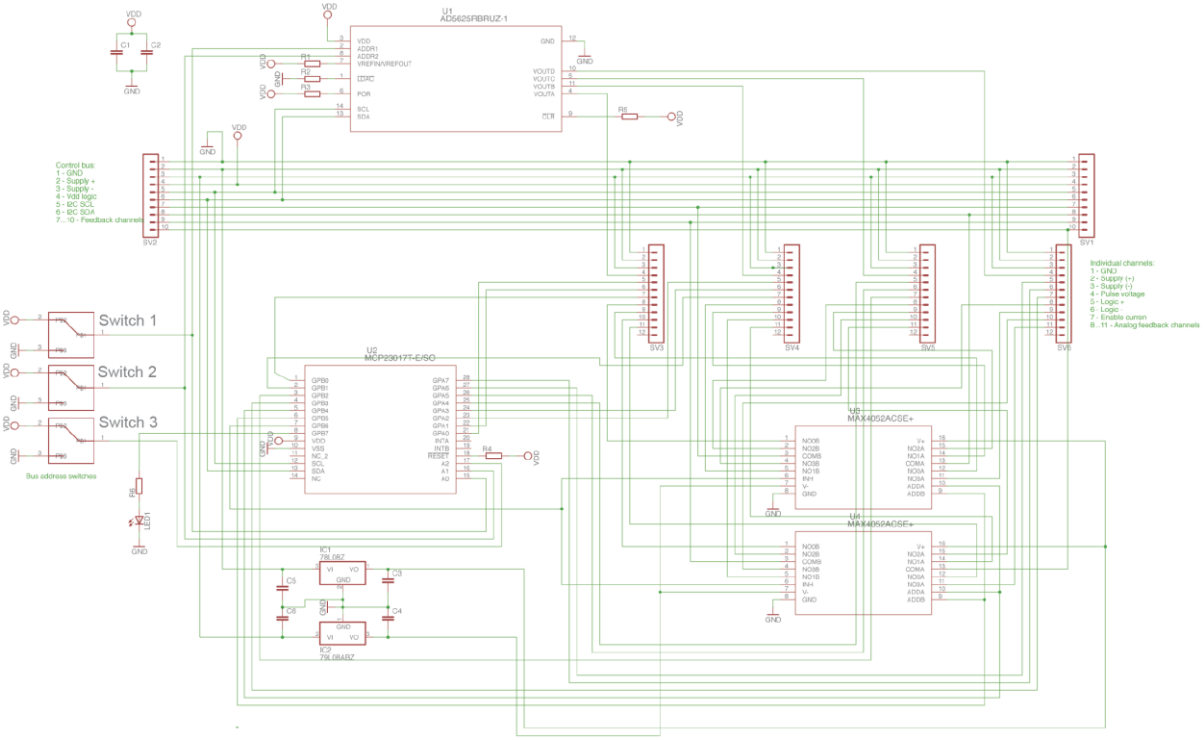
Schematic symbol	Value	Description	Product code (Digikey)
SV1	90degree 12-pin male 0.1" pin header	Connecting the amplifier board to the multiplexer board	609-3330-ND
IC1, IC2	AD820R	Operational amplifiers. IC1 generates the pulse shape and IC2 drives the current through the stimulation electrodes, such that current corresponds to the input voltage $I = V_{in}/R_9$	AD820ARZ-ND
U1	TLP176A	MOSFET optocoupler, which enables or disables the current through the electrodes	TLP176AM(TPLECT-ND
Switch 1, Switch 2	Small two position switches	Switch 1 and 2 configure the feedback mode of the amplifier IC2. It can be constant current mode (switch 1 is 1→3 and switch 2 is 1→2 positions) which is used throughout this work here. Other optional configuration is constant field mode (switch 1 is 1→2 and switch 2 is 1→3 positions) which requires additional sensing electrode to be included.	401-2016-1-ND
J1	Barrel connector	Connector for the stimulator electrodes. It has 4 positions, which can be adapted for both 2 and 4 electrode configurations	839-1412-ND
T1, T2	BC63017	NPN transistors for defining pulse polarity	MMBT3904TPMSCT-ND
R1, R2	12kΩ	Input resistors for transistors base	A129762CT-ND
R3, R4	1.2kΩ	Pull-up resistors for transistor collector	P1.20KCCT-ND
R5, R6, R7, R8	120kΩ	Pulse shape formation network	P120KCCT-ND
R9	330Ω	Gain resistor defining the electrode current and thus also maximum field strength	P330CCT-ND
R18	1kΩ	Optocouple input resistor	RHM1.00KAECT-ND
C1, C2	100pF	Filtering capacitors	732-12228-1-ND
C3, C4	10µF	Supply stabilizing capacitors	1276-6455-1-ND
		Optional components for alternative configurations	

R10-R17	100kΩ	Feedback to monitor electrode potential over maximum of 4 electrodes per channel	311-100KCRCT-ND

Supplementary Table 2. Input and output description of the channel amplifier board.

<table>
<tr><th>Connector</th><th>Pin</th><th>Description</th></tr>
<tr><td rowspan="11">SV1</td><td>1</td><td>Ground</td></tr>
<tr><td>2</td><td>Positive supply voltage +15V</td></tr>
<tr><td>3</td><td>Negative supply voltage -15V</td></tr>
<tr><td>4</td><td>Pulse amplitude control. Analog signal V_{in} with range 0 to +3.3V. Electrode current would be $I = V_{in}/R_9$</td></tr>
<tr><td>5</td><td>Pulse polarity pin “VD+” (digital signal 0 or +3.3V). For further description see next row.</td></tr>
<tr><td>6</td><td>Pulse polarity pin “VD-” (digital signal 0 or +3.3V)
Since R3 and R4 are much smaller than R5=R6=R7=R8, we can neglect R3 and R4. In this case resistors R6 and R7 can have left terminals biased either 0, when corresponding polarity pin is high (´1´) or V_{in} when pin is low (´0´). Thus the output of amplifier IC1 becomes
<table>
<tr><th>VD+</th><th>VD-</th><th>Output of IC1</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>$-V_{in}$</td></tr>
<tr><td>1</td><td>0</td><td>V_{in}</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>7</td><td>Optocoupler enable pin (digital signal 0 or +3.3V)
If high, current is enabled through the electrodes, if low, the electrode output is in high-impedance state (no current)</td></tr>
<tr><td>8,9,10,11</td><td>Electrode potential monitoring pins for other optional configurations (not used here)</td></tr>
<tr><td>12</td><td>Reserved / Not connected</td></tr>
<tr><td rowspan="4">J1</td><td>1</td><td>Working electrode 1 (current driving)</td></tr>
<tr><td>2</td><td>Working electrode 2 (current driving)</td></tr>
<tr><td>3</td><td>Sensing electrode 1 (potential sensing, optional configuration, not used here)</td></tr>
<tr><td>4</td><td>Sensing electrode 2 (potential sensing, optional configuration, not used here)</td></tr>
</table>

1.2. Multiplexer board



Supplementary Figure 2. Circuit diagram of the multiplexer board. One multiplexer board drives 4 amplifier boards connected to it. Multiplexer boards have switches to assign each board individual address, which allows them to be stacked in series and control through a common bus (I2C).

Supplementary Table 3. Components of the multiplexer board.

<table>
<tr><th>Schematic symbol</th><th>Value</th><th>Description</th><th>Product code (Digikey)</th></tr>
<tr><td>SV1, SV2</td><td>90degree 10-pin 0.1” pin headers one male and one female</td><td>These connectors allow to stack up to 4 multiplexer boards in series (current system uses 3 boards)</td><td>S1111EC-10-ND

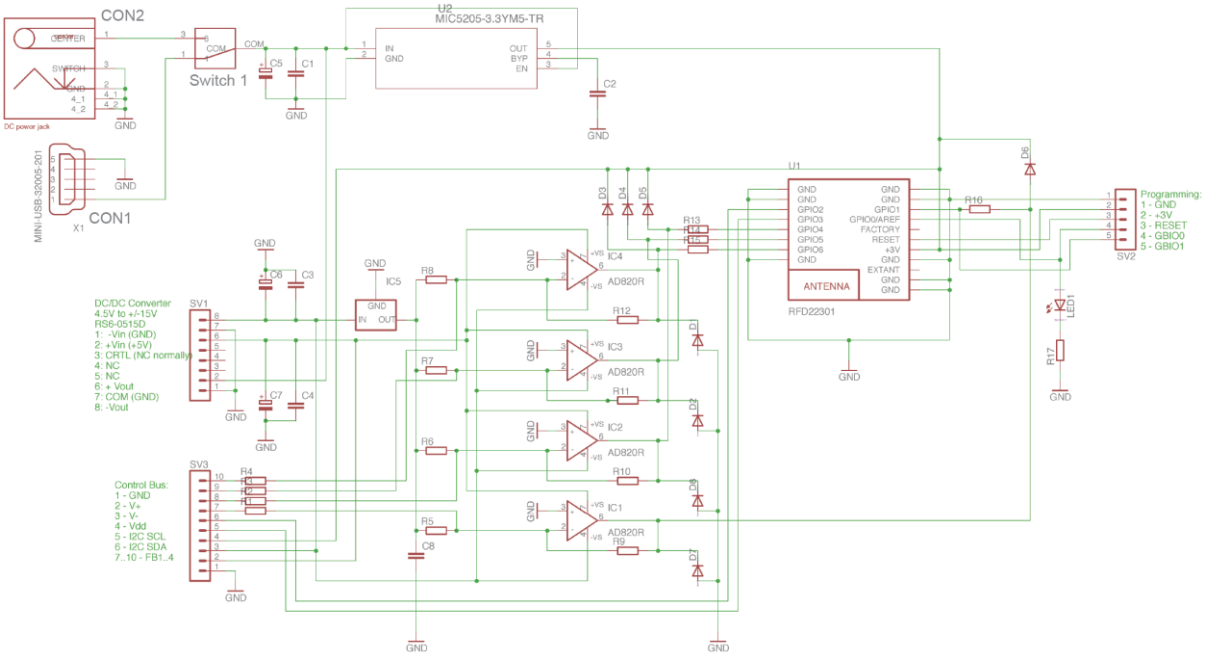
S5446-ND</td></tr>
<tr><td>SV3,SV4, SV5, SV6</td><td>12-pin straight female 0.1” pin header</td><td>For connecting individual amplifier boards</td><td>609-3553-ND</td></tr>
<tr><td>U1</td><td>AD5625RBRUZ</td><td>4 channel 16-bit digital-to-analog converter (DAC) with I2C interface</td><td>AD5625RBRUZ-1-ND</td></tr>
<tr><td>U2</td><td>MCP23017</td><td>16-bit IO port expander with I2C interface</td><td>MCP23017T-E/SOCT-ND</td></tr>
<tr><td>LED1</td><td>Blue LED</td><td>Indicator LED (blinking shows proper communication)</td><td>VAOL-S8SB4CT-ND</td></tr>
<tr><td>Switch 1, Switch 2, Switch3</td><td>Small two position switches</td><td>These allow to configure individual I2C addresses for each of the multiplexer boards, each switch can be connected to ground (GND, logic ‘0’) of digital supply (Vdd, logic ‘1’). Switch 3 is in this system always connected to Vdd. Each board has unique combinations of Switch 1 and 2 states, defining I2C addresses (in HEX) as following:
<table>
<tr><th>Switch 1</th><th>Switch 2</th><th>MCP address</th><th>DAC address</th></tr>
<tr><td>0</td><td>0</td><td>0x24</td><td>0x1F</td></tr>
<tr><td>0</td><td>1</td><td>0x25</td><td>0x1C</td></tr>
<tr><td>1</td><td>0</td><td>0x26</td><td>0x13</td></tr>
<tr><td>1</td><td>1</td><td>0x27</td><td>0x10</td></tr>
</table></td><td>401-2016-1-ND</td></tr>
<tr><td>R1, R2, R3, R4, R5</td><td>10kΩ</td><td colspan="2">Biasing control pins of ICs</td><td>A126332CT-ND</td></tr>
<tr><td>R6</td><td>100Ω</td><td colspan="2">LED resistor</td><td>311-100CRCT-ND</td></tr>
<tr><td>C1</td><td>1µF</td><td colspan="2">Digital supply stabilizing capacitor</td><td>399-1284-1-ND</td></tr>
<tr><td>C2</td><td>10µF</td><td colspan="2">Digital supply stabilizing capacitor</td><td>1276-6455-1-ND</td></tr>
<tr><td></td><td></td><td colspan="2">Optional components for alternative configurations</td><td></td></tr>
<tr><td>U3, U4</td><td>MAX4052ACS</td><td colspan="2">Analog switches for selection of potential feedback inputs</td><td>MAX4052ACSE+-ND</td></tr>
</table>

IC1	78L08Z	Positive linear voltage regulator +8V (For U3 and U4)	497-10099-1-ND
IC2	79L08Z	Negative linear voltage regulator -8V (For U3 and U4)	497-7299-1-ND
C3, C4, C5, C6	10µF	Supply stabilizing capacitors	

Supplementary Table 4. Input and output description of the multiplexer board (common bus)

Pin	Description
1	Ground
2	Positive supply voltage +15V
3	Negative supply voltage -15V
4	Digital circuits supply voltage +3.3V
5	I2C bus clock signal (SCL)
6	I2C bus data signal (SDA)
7,8,9,10	Electrode potential monitoring pins for other optional configurations (not used here)

1.3. Motherboard



Supplementary Figure 3. Circuit diagram of the motherboard. Mother board contains Arduino microcontroller, power supply and interface to the multiplexer boards. Additional optional features would allow implementation of potential monitoring on up to 4 electrodes, but are not used here.

Supplementary Table 5. Components of the motherboard.

Schematic symbol	Value	Description	Product code (Digikey)
SV1	DC/DC converter (RS6-0515D)	Connector for DC/DC converter from 4.5V to +/-15V	945-2979-ND
SV2	5-pin 0.1” pin header	Connector for programming to load firmware (maybe not required depending on the choice of Arduino board)	SAM1184-05-ND
SV3	90degree 10-pin 0.1” pin headers female	Connector to the first multiplexer board	S5446-ND
U1	RFD22301	Arduino compatible microcontroller module. Can be substituted with any Arduino with I2C interface and 1 digital output (optional configuration may require additional 4 analog inputs)	1562-1000-ND
U2	MIC5205-3.3YM5	Positive linear voltage regulator (LDO) with +3.3V output, for supplying digital ICs.	576-1259-1-ND
Switch 1	Two state switch	Power switch	GH7540CT-ND
CON1	Mini-USB connector	Power input from USB charger	151-1206-1-ND

LED1	Blue LED	Indicator LED for microcontroller	VAOL-S8SB4CT-ND
R17	100Ω	LED resistor	311-100CRCT-ND
C1, C3, C4	1µF	Supply power stabilization capacitors	399-1284-1-ND
C2	420pF	U2 stability capacitor	1276-2516-1-ND
C5, C6, C7	100µF	Supply power stabilization capacitors	P5182-ND
		Optional components for alternative configurations	
IC1, IC2, IC3, IC4	AD820R	Operational amplifiers for electrode potential inputs	AD820ARZ-ND
IC5	-5V LDO	Negative linear voltage regulator with -5V output	497-2960-ND
D1-D8	Signal diode	Microcontroller board analog signal input protection	LL4148FSCT-ND
R1, R2, R3, R4	1MΩ	Input resistors for potential monitoring inputs	A106058CT-ND
R5, R6, R7, R8	604kΩ	Input resistors for potential monitoring inputs	311-604KCRCT-ND
R9, R10, R11, R12	180kΩ	Operational amplifiers (IC1-IC4) feedback resistors	P180KCCT-ND
R13, R14, R15, R16	100kΩ	Microcontroller board analog input interface (might need to be adjusted for different Arduino, depending on input impedance)	311-100KCRCT-ND
C8	1µF	Supply power stabilization capacitors	399-1284-1-ND
CON2		Alternative power connector	CP-032HPJCT-ND

Supplementary Table 6. Additional electronics components external to the controller.

Description	Product code (Digikey)
Cable to connect electrodes on the lid and the controller	T1245-1-ND
USB wall adapter for powering the device	1470-4188-ND
USB cable to connect the adapter and controller	380-1428-ND

2. Device cost

Supplementary Table 7. Cost of components to construct the stimulator system. Cost is given for volumes corresponding single unit. Cost would be reduced significantly for higher volumes, especially for printed circuit boards and electronic components, also titanium foil is sufficient to build numerous device and lower cost alternatives exist.

Description	Cost (EUR)
Printed circuit boards (Eurocircuits)*	210.68
Electronic components for presented configuration (Digikey)*	344.6
Platinized titanium foil 10x10cm (Goodfellow: 386-456-57)**	571
Transparent acrylic sheet 20x20cm (Amazon.es: B07PK7BTDJ)	13.21
TOTAL	**1140**

* as of November 2018, ** as of August 2019

3. Firmware

Firmware was written in Arduino platform using only general functions and Wire library for I2C and can be therefore adapted also to other Arduinos. Wire function for pin assignment may need to be modified depending on the particular controller. Also timing shall be calibrated due to different execution time depending on the speed of the processor. Below is provided one exemplary firmware code, which results in pulse protocols described in table S8. In presented experiments different pulse protocols and firmware were used. In order to modify pulse protocols, function `initOutputArrays()` would be adjusted to define pulse timing and `initDACs()` for setting field strengths for each channel. In current implementation pulse protocols can have period of 500 steps (=500ms). Since the pulse sequence is controlled by the MCP23017 (IO expander ICs), each state is described by the 2 byte output code per one multiplex board and MCP chip. Since the device has three multiplexer boards six (2x3) 500 element char (byte) arrays `OUT_A1[500]` … `OUT_B3[500]` are used to store all the signal shapes and are initialized during the start-up of the controlled by `initOutputArrays()` function, thereafter values from these

arrays are fetched periodically and loaded into MCP chips in the main loop() function. For each of the steps, the current can be enabled or disabled and polarity can be defined, however field strength (pulse amplitude) of the pulse is kept constant for one channel through the operation. Firmware program starts, when Arduino is powered and does not require any user input. Therefore system is also safe in case of a temporal power loss and restart and protocol resumes as soon as the power is restored. It should be noted that switching on and off the optocoupler costs additional 0.5ms, thus reducing 1ms pulse to 0.5ms pulse and 2ms pulse to 1.5ms pulse etc. In case optocoupler is not switched, one pulse step corresponds to 1ms duration.

Supplementary Table 8. Experimental parameters implemented by the demo firmware below.

Channel	Well	Field (E/m)	Pulse duration (t_p) (ms)	Delay (t_1) (ms)	Delay (t_2) (ms)	Frequency (Hz)
1	A1	25	1	0	0	500
2	A2	25	10	0	0	50
3	A3	25	125	0	0	4
4	A4	150	10	0	0	50
5	B1	75	10	0	0	50
6	B2	10	10	0	0	50
7	B3	150	0.5	1.5	2.5	200
8	B4	150	0.5	1.5	47.5	20
9	C1	150	0.5	1.5	497.5	2
10	C2	150	1.5	1.5	45.5	20
11	C3	150	4.5	1.5	39.5	20
12	C4	0	Control (no stimulation)			

Supplementary Firmware Code

```
// === In-vitro electrical stimulator for cell cultures ===
// ----------------------------------------------------------------------------
// -- It contains following features:
// One "Motherboard" interactiong with up to four different "Multiplexer boards" (currently 3)
// which are interfaced over I2C bus.
// Each Multiplexer board contains switches to set their address:
// Switch 1 | Switch 2 | MCP  | DAC  | Board
//-------------------------------------------------------------
// 1       | 1       | 0x27 | 0x10 | 0
// 0       | 1       | 0x25 | 0x1C | 1
// 1       | 0       | 0x26 | 0x13 | 2
// 0       | 0       | 0x24 | 0x1F | 3
//-------------------------------------------------------------
// Switch 3 is set to 1 on all boards.
// Each DAC controls four channels
// Channels from left to right are 0 to 3
// 0 to 3 are VOUT A to D
// MCP controls other functions with following mapping
// PORT A
// Port MCP A.0 -> Ch0 (Logic +)
// Port MCP A.1 -> Ch0 (Logic -)
// Port MCP A.2 -> Ch1 (Logic +)
// Port MCP A.3 -> Ch1 (Logic -)
// Port MCP A.4 -> Ch2 (Logic +)
// Port MCP A.5 -> Ch2 (Logic -)
// Port MCP A.6 -> Ch3 (Logic +)
// Port MCP A.7 -> Ch3 (Logic -)
// PORT B
// Port MCP B.0 -> Ch0 Enable
// Port MCP B.1 -> Ch1 Enable
// Port MCP B.2 -> Ch2 Enable
// Port MCP B.3 -> Ch3 Enable
// Port MCP B.4 -> MAX4052 ADDA
// Port MCP B.5 -> MAX4052 ADDB
// Port MCP B.6 -> MAX4052 INH (All switches open, if '1')
// Port MCP B.7 -> LED on Multiplexer board
```

```
// Channel selection map is:
// ADDA:0, ADDB:0 -> Ch0
// ADDA:1, ADDB:0 -> Ch1
// ADDA:0, ADDB:1 -> Ch2
// ADDA:1, ADDB:1 -> Ch3
// ------------------------------------------------------------
// MCP23017 I2C to Parallel converter, which controlls
//  2x 8-to-1 MUXes, 1x 32-to-1 MUX, and 2x indicator LED
// AD5625 I2C 4-ch nanoDAC
// These are controlled over I2C. MCP23017 has address 0x20
// AD5625 has address 0x1F
// Analog input on GPIO6
// It works with Arduino (currently used device is RFDUINO)
// ---
// INL - International Iberian Nanotechnology Laboratory
// Braga, Portugal
// ------------------------------------------------------------

const char MCP_ADD[4] = {0x27, 0x25, 0x26, 0x24}; //MCP addresses
const char DAC_ADD[4] = {0x10, 0x1C, 0x13, 0x1F}; //DAC addresses
const char AIN[4]= {1, 4, 5, 6}; // Analog channels

//Internal registers for Pins
//These are the configuration pins for the each multiplexer board
char pinsA[4] = {0, 0, 0, 0};
char pinsB[4] = {128, 128, 128, 128};

#include <Wire.h>
// I2C pin configuration (Some Arduinos have fixed I2C mapping)
#define SDApin 2
#define SCLpin 3

//------------------------------------------------------------------------------
// MCP I2C to PARALLEL CONVERTER FUNCTIONS
//  setupMCP    - initialize
//  sendToMCP   - send raw bytes
//  setMCPPortA - set value of channel A (8-bit)
//  setMCPPortB - set value of channel B (8-bit)
//------------------------------------------------------------------------------

//This sets up MCP23017
//board is the number from 0 to 3
void setupMCP(byte board)
{
 sendToMCP(MCP_ADD[board], 0x00, 0x00); //Port A is output
 sendToMCP(MCP_ADD[board], 0x01, 0x00); //Port B is output
 sendToMCP(MCP_ADD[board], 0x14, 0x00); //Port A is 0
 sendToMCP(MCP_ADD[board], 0x15, 0x00); //Port B is 0
}

// This sends physical command to MCP23017
// INPUTS:
// add: is address of the device
// regadd: register address byte
// val: register value
// We are only writing MCP is only in out mode.
void sendToMCP(byte add, byte regadd, byte val)
{
 byte error;
 Wire.beginTransmission(add); //Device OP code
 Wire.write(regadd); Wire.write(val);
 error = Wire.endTransmission();
}

//Set MCP Port A
//Input is 8-bit port value
void setMCPPortA(byte board, byte val)
{
 sendToMCP(MCP_ADD[board], 0x14, val);
}
```

```
//Set MCP Port B
//Input is 8-bit port value
void setMCPPortB(byte board, byte val)
{
 sendToMCP(MCP_ADD[board], 0x15, val);
}

//Update port setting now physically
void updatePORT(byte board)
{
   setMCPPortA(board,pinsA[board]);
   setMCPPortB(board,pinsB[board]);
}

//--------------------------------------------------------------------------
// DAC FUNCTIONS
//  setupDAC  - initialize
//  setDAC    - set value of a channel
//  sendToDAC - send raw bytes
//--------------------------------------------------------------------------

//This function configures nanoDAC to be ready to use
//Board is the board number of DAC to initialize
void setupDAC(byte board, bool internalref)
{
 sendToDAC(DAC_ADD[board], 0x20, 0x00, 0x0F); //First command set DAC output into normal mode
 if (internalref) {
   sendToDAC(DAC_ADD[board], 0x38, 0x00, 0x01);  //Use internal reference
 }
 else {
   sendToDAC(DAC_ADD[board], 0x38, 0x00, 0x00);  //Use external reference
 }
 sendToDAC(DAC_ADD[board], 0x30, 0x00, 0x0F); //Don´t use LDAC, update output by software
}

//This functions updates one DAC channel
//first byte is the board number 0 to 3
//byte is the DAC channel from 0 to 3 (Corresponding to A..D)
//Value is in the range:
//Total time this function excecutes is about 150us. (update time)
void setDAC(byte board, byte channel, unsigned int value)
{
 byte lowB = value & 0xFF;
 byte highB = (value >> 8) & 0xFF;
 byte chB = 0x18 + channel;
 sendToDAC(DAC_ADD[board], chB, highB, lowB);
}

//This sends physical command to DAC.
//First is the dac address
void sendToDAC(byte add, byte b1, byte b2, byte b3)
{
 byte error;
 Wire.beginTransmission(add);
 Wire.write(b1); Wire.write(b2); Wire.write(b3);
 error = Wire.endTransmission();
}

//Here we set the DAC voltages, these are not gonna change during operation
//in this version
//Define electric fields used. This is calculated from calibration informations
#define E25 7385
#define E150 42114
#define E75 21277
#define E10 3218

void initDACs()
{
 //Setup DAC
 setupDAC(0, true); // We use internal reference.
 setupDAC(1, true); // We use internal reference.
```

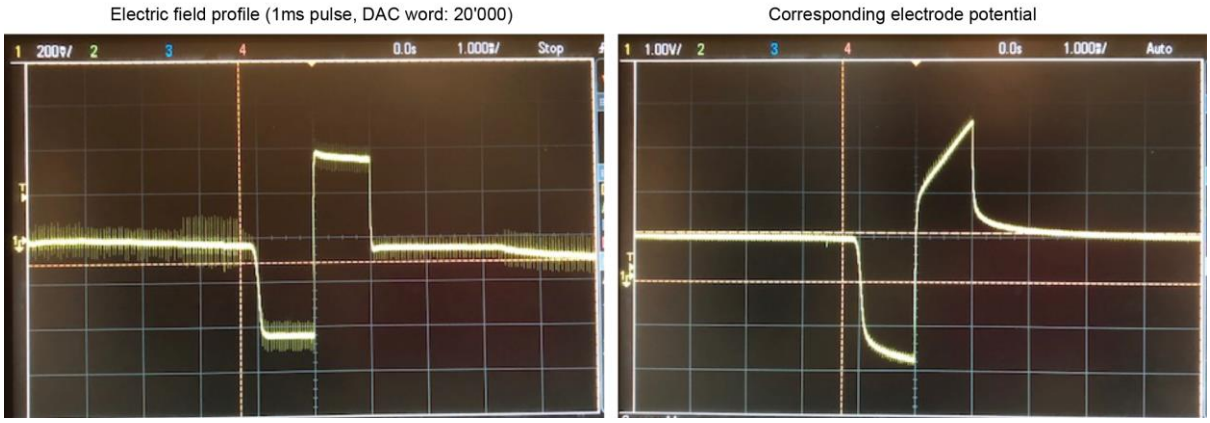
```
 setupDAC(2, true); // We use internal reference.
 //Set Values
 setDAC(0,0,E25);  //1:  A1: 25
 setDAC(0,1,E25);  //2:  A2: 25
 setDAC(0,2,E25);  //3:  A3: 25
 setDAC(0,3,E150); //4:  A4: 150

 setDAC(1,0,E75);  //5:  B1: 75
 setDAC(1,1,E10);  //6:  B2: 10
 setDAC(1,2,E150); //7:  B3: 150
 setDAC(1,3,E150); //8:  B4: 150

 setDAC(2,0,E150);  //9:  C1: 150
 setDAC(2,1,E150);  //10: C2: 150
 setDAC(2,2,E150); //11: C3: 150
 setDAC(2,3,0);    //12: C4: 0
}

//Here we setup arrays for the pulse sequences
//Arrays are not gonna change during the operation
//We just loop through them continuously
//Total cycle is 500ms (500steps)
//For simplicity we make 6 arrays
//Each array represent states of A or B port
//on the multiplexer board
char OUT_A1[500];
char OUT_B1[500];
char OUT_A2[500];
char OUT_B2[500];
char OUT_A3[500];
char OUT_B3[500];

// THIS FUNCTION DEFINES THE PULSE SEQUENCE
void initOutputArrays()
{
 //int cT; //Cycle period
 //First it is all 0x00
 for(int i=0; i<500; i++)
 {
  OUT_A1[i]=0x00;
  OUT_B1[i]=0x00;
  OUT_A2[i]=0x00;
  OUT_B2[i]=0x00;
  OUT_A3[i]=0x00;
  OUT_B3[i]=0x00;
 }

 //Each channel go through
 //- 1: A1: 1ms:1ms-->2ms pulses always on
 for(int i=0; i<250; i++)
 {
  OUT_A1[i*2]=OUT_A1[i*2]|0x01;
  OUT_A1[i*2+1]=OUT_A1[i*2+1]|0x02;
  OUT_B1[i*2]=OUT_B1[i*2]|0x01;
  OUT_B1[i*2+1]=OUT_B1[i*2+1]|0x01;
 }
 //- 2: A2: 10ms:10ms-->20ms pulses always on
 for(int i=0; i<25; i++)
 {
  for(int j=0;j<10;j++)
  {
   OUT_A1[i*20+j]=OUT_A1[i*20+j]|0x04;
   OUT_A1[i*20+j+10]=OUT_A1[i*20+j+10]|0x08;
   OUT_B1[i*20+j]=OUT_B1[i*20+j]|0x02;
   OUT_B1[i*20+j+10]=OUT_B1[i*20+j+10]|0x02;
  }
 }
 //- 3: A3: 125ms:125ms-->250ms pulses always on
 for(int i=0; i<2; i++)
 {
  for(int j=0;j<125;j++)
```

```
  {
   OUT_A1[i*250+j]=OUT_A1[i*250+j]|0x10;
   OUT_A1[i*250+j+125]=OUT_A1[i*250+j+125]|0x20;
   OUT_B1[i*250]=OUT_B1[i*250+j]|0x04;
   OUT_B1[i*250+j+125]=OUT_B1[i*250+j+125]|0x04;
  }
 }
 //- 4: A4: 10ms:10ms-->20ms pulses always on
 for(int i=0; i<25; i++)
 {
  for(int j=0;j<10;j++)
  {
   OUT_A1[i*20+j]=OUT_A1[i*20+j]|0x40;
   OUT_A1[i*20+j+10]=OUT_A1[i*20+j+10]|0x80;
   OUT_B1[i*20+j]=OUT_B1[i*20+j]|0x08;
   OUT_B1[i*20+j+10]=OUT_B1[i*20+j+10]|0x08;
  }
 }
 //-----------------------------------------------
 //- 5: B1: 10ms:10ms-->20ms pulses always on
 for(int i=0; i<25; i++)
 {
  for(int j=0;j<10;j++)
  {
   OUT_A2[i*20+j]=OUT_A2[i*20+j]|0x01;
   OUT_A2[i*20+j+10]=OUT_A2[i*20+j+10]|0x02;
   OUT_B2[i*20+j]=OUT_B2[i*20+j]|0x01;
   OUT_B2[i*20+j+10]=OUT_B2[i*20+j+10]|0x01;
  }
 }
 //- 6: B2: 10ms:10ms-->20ms pulses always on
 for(int i=0; i<25; i++)
 {
  for(int j=0;j<10;j++)
  {
   OUT_A2[i*20+j]=OUT_A2[i*20+j]|0x04;
   OUT_A2[i*20+j+10]=OUT_A2[i*20+j+10]|0x08;
   OUT_B2[i*20+j]=OUT_B2[i*20+j]|0x02;
   OUT_B2[i*20+j+10]=OUT_B2[i*20+j+10]|0x02;
  }
 }
 //- 7: B3: 1ms:1ms:1ms:2ms-->5ms pulses always on
 for(int i=0; i<100; i++)
 {
   OUT_A2[i*5+0]=OUT_A2[i*5+0]|0x10; //1
   OUT_B2[i*5+0]=OUT_B2[i*5+0]|0x04;
   OUT_A2[i*5+1]=OUT_A2[i*5+1]|0x00; //2
   OUT_B2[i*5+1]=OUT_B2[i*5+1]|0x00;
   OUT_A2[i*5+2]=OUT_A2[i*5+2]|0x20; //3
   OUT_B2[i*5+2]=OUT_B2[i*5+2]|0x04;
   OUT_A2[i*5+3]=OUT_A2[i*5+3]|0x00; //4
   OUT_B2[i*5+3]=OUT_B2[i*5+3]|0x00;
   OUT_A2[i*5+4]=OUT_A2[i*5+4]|0x00; //5
   OUT_B2[i*5+4]=OUT_B2[i*5+4]|0x00;
 }
 //-8: B4: 1ms:1ms:1ms:47ms-->50ms pulses always on
 for(int i=0; i<10; i++)
 {
   OUT_A2[i*50+0]=OUT_A2[i*50+0]|0x40; //1
   OUT_B2[i*50+0]=OUT_B2[i*50+0]|0x08;
   OUT_A2[i*50+1]=OUT_A2[i*50+1]|0x00; //2
   OUT_B2[i*50+1]=OUT_B2[i*50+1]|0x00;
   OUT_A2[i*50+2]=OUT_A2[i*50+2]|0x80; //3
   OUT_B2[i*50+2]=OUT_B2[i*50+2]|0x08;
   for(int j=0; j<47; j++)
   {
    OUT_A2[i*50+3+j]=OUT_A2[i*50+3+j]|0x00; //4-50
    OUT_B2[i*50+3+j]=OUT_B2[i*50+3+j]|0x00;
   }
  }
 //-----------------------------------------------
```

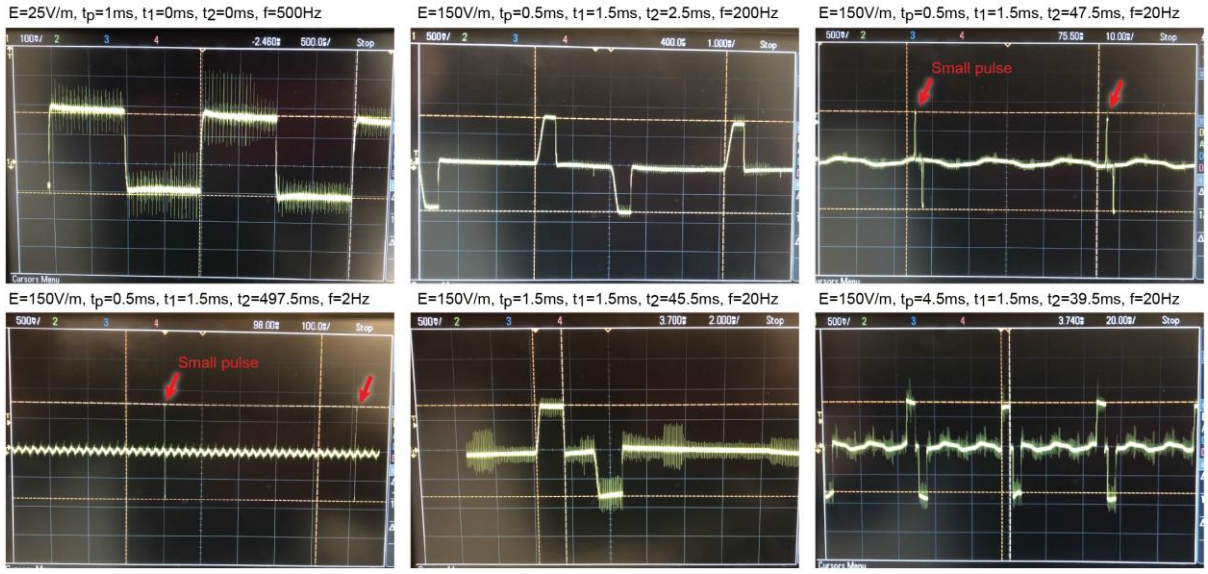
```
  //-9: C1: 1ms:1ms:1ms:497ms-->500ms pulses always on
     OUT_A3[0]=OUT_A3[0]|0x01; //1
     OUT_B3[0]=OUT_B3[0]|0x01;
     OUT_A3[1]=OUT_A3[1]|0x00; //2
     OUT_B3[1]=OUT_B3[1]|0x00;
     OUT_A3[2]=OUT_A3[2]|0x02; //3
     OUT_B3[2]=OUT_B3[2]|0x01;
     for(int j=0; j<497; j++)
     {
       OUT_A3[3+j]=OUT_A3[3+j]|0x00; //4-50
       OUT_B3[3+j]=OUT_B3[3+j]|0x00;
     }
  //-10: C2: 2ms:1ms:2ms:45ms-->50ms pulses always on
  for(int i=0; i<10; i++)
  {
     OUT_A3[i*50+0]=OUT_A3[i*50+0]|0x04; //1
     OUT_B3[i*50+0]=OUT_B3[i*50+0]|0x02;
     OUT_A3[i*50+1]=OUT_A3[i*50+1]|0x04; //2
     OUT_B3[i*50+1]=OUT_B3[i*50+1]|0x02;
     OUT_A3[i*50+2]=OUT_A3[i*50+2]|0x00; //3
     OUT_B3[i*50+2]=OUT_B3[i*50+2]|0x00;
     OUT_A3[i*50+3]=OUT_A3[i*50+3]|0x08; //4
     OUT_B3[i*50+3]=OUT_B3[i*50+3]|0x02;
     OUT_A3[i*50+4]=OUT_A3[i*50+4]|0x08; //5
     OUT_B3[i*50+4]=OUT_B3[i*50+4]|0x02;
     for(int j=0; j<45; j++)
     {
       OUT_A3[i*50+5+j]=OUT_A3[i*50+5+j]|0x00; //4-50
       OUT_B3[i*50+5+j]=OUT_B3[i*50+5+j]|0x00;
     }
  }
  //-11: C3: 5:1:5:39-->50ms
  for(int i=0; i<10; i++)
  {
     for(int j=0; j<5; j++)
     {
       OUT_A3[i*50+j]=OUT_A3[i*50+j]|0x10; //1-5
       OUT_B3[i*50+j]=OUT_B3[i*50+j]|0x04;
     }
     OUT_A3[i*50+5]=OUT_A3[i*50+5]|0x00; //1
     OUT_B3[i*50+5]=OUT_B3[i*50+5]|0x00;
     for(int j=0; j<5; j++)
     {
       OUT_A3[i*50+6+j]=OUT_A3[i*50+6+j]|0x20; //1-5
       OUT_B3[i*50+6+j]=OUT_B3[i*50+6+j]|0x04;
     }
     for(int j=0; j<39; j++)
     {
       OUT_A3[i*50+11+j]=OUT_A3[i*50+11+j]|0x00; //4-50
       OUT_B3[i*50+11+j]=OUT_B3[i*50+11+j]|0x00;
     }
   }
   //*/
  //LED Blinking
  for(int i=0;i<160;i++)
  {
    OUT_B1[i]=OUT_B1[i]|0x80;
    OUT_B2[i+160]=OUT_B2[i+160]|0x80;
    OUT_B3[i+320]=OUT_B3[i+320]|0x80;
  }
}

// Setup microcontroller
void setup()
{
  //Setup I2C bus
  Wire.speed = 400; //400; //Speed
  Wire.begin();
  Wire.beginOnPins(SCLpin, SDApin);
  //Serial
  Serial.begin(9600);
```

```
  Serial.println("Welcome to in-vitro stimulator");
  initOutputArrays();
  initDACs();
  setupMCP(0);
  setupMCP(1);
  setupMCP(2);
  updatePORT(0);
  updatePORT(1);
  updatePORT(2);
}

// --- Main loop. We can compile and test these different options here
int counter=0;
void loop()
{
    counter=(counter+1)%500;
    pinsA[0]=OUT_A1[counter];
    pinsA[1]=OUT_A2[counter];
    pinsA[2]=OUT_A3[counter];
    pinsB[0]=OUT_B1[counter];
    pinsB[1]=OUT_B2[counter];
    pinsB[2]=OUT_B3[counter];
    updatePORT(0);
    updatePORT(1);
    updatePORT(2);
    //Necessary delay might be needed here
    //To right update frequency
}

// --- END OF CODE ---
```

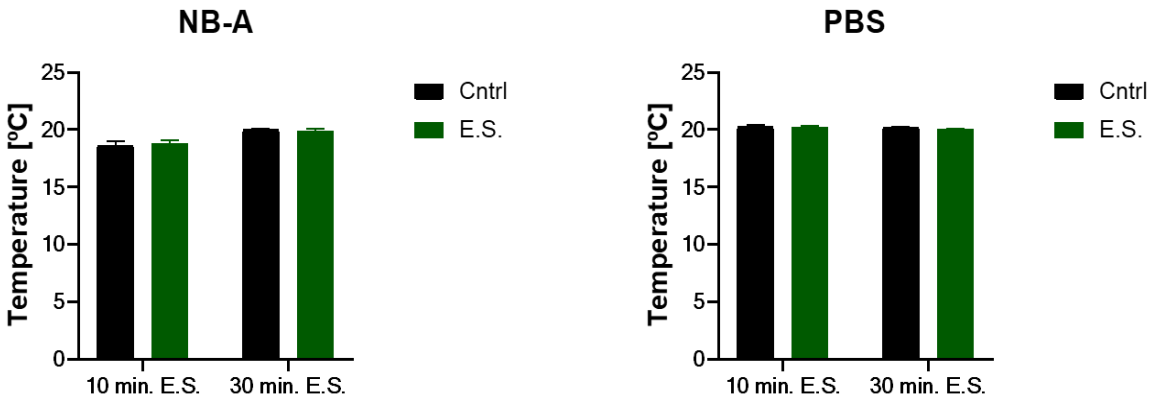
4. Additional test data



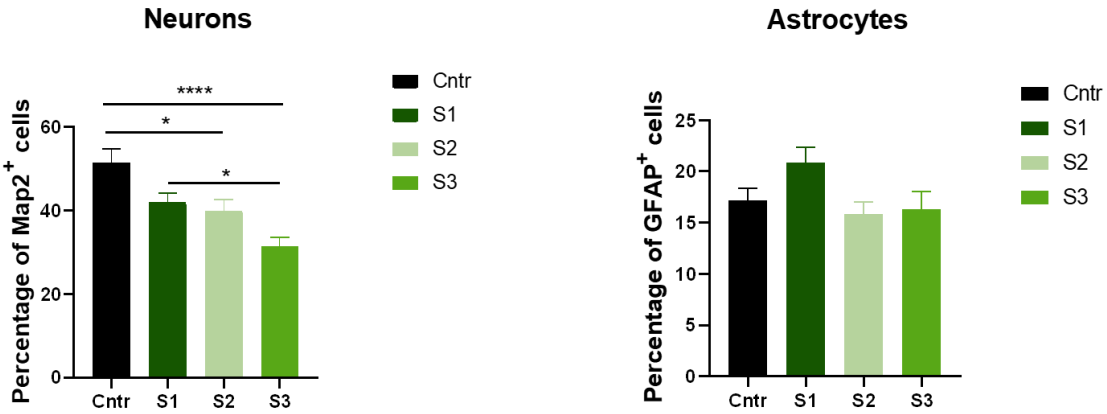
Supplementary Figure 4. Example of electric field profile in current controlled mode (on left) and corresponding electrical potential on the stimulation electrodes (on right), as it can be seen voltage varies during the pulse, which is due to electrode capacitance time dependent nature of electrochemical reactions, therefore current controlled mode ensures well defined field profile.



Supplementary Figure 5. Further examples of electric field profiles as measured with oscilloscope for different settings. Some noise in the measured data is observed due to small sensing electrodes and unshielded measurement cables. Observe different vertical and horizontal scales on different panels.



Supplementary Figure 6. Temperature Readings after Electrical Stimulation. Temperatures after 10 and 30 minutes of E.S. on NB-A and PBS medium. n = 4, data are show as mean ±SEM.



Supplementary Figure 7. Percentage of neurons and astrocytes. Percentage of neurons and astrocytes in spinal cord mixed cultures after different stimulations protocols. S2 and S3 settings significantly reduced the percentage of neurons. Nome of the stimulation protocols influence the percentage of astrocytes. Data are show as mean ±SEM. *p value < 0,05; ****p value < 0,0001.