Supplementary Information

Design of a Low-Cost and High-Precision Measurement System Suitable for Organic Transistors

Vratislav Režo [1], Martin Weis [1]

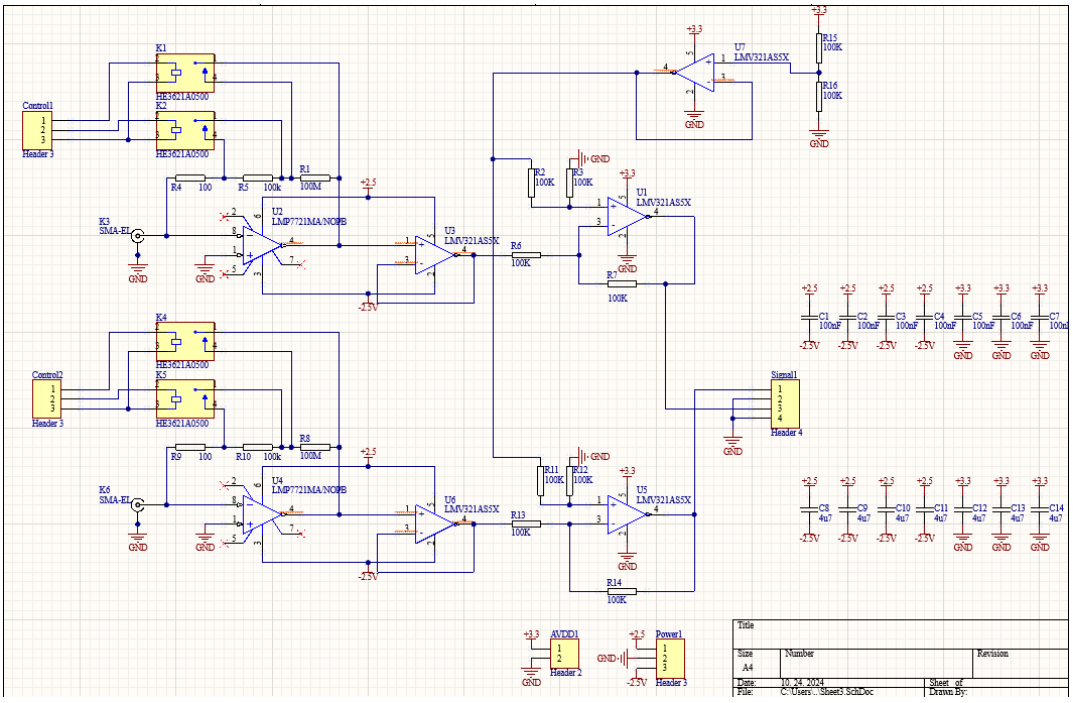
[1] Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, 812 43 Bratislava, Slovakia; vratislav.rezo@stuba.sk

Table S1. Comparison of various previously reported approaches.

Feature	Simple shunt resistor	Shunt resistor with buffer amplifier	Resistive TIA	Capacitive TIA	Current amplifier
Operating principle	Voltage drop across resistor measured.	Buffer minimizes loading from shunt resistor.	Resistor converts current to voltage.	Current integrated on capacitor, then differentiated.	Current amplified directly.
Advantages	Simple, low cost.	High input impedance, voltage output.	Simple, DC measurement.	High bandwidth, low noise, stable gain.	High input impedance, no added noise from later stages.
Disadvantages	Loading effects, high noise for small currents.	Buffer errors, more complex than simple shunt.	Limited bandwidth, high noise for small currents.	Integrator saturation, complex design, frequency-dependent range.	Shot noise limits sensitivity.
Specific techniques for performance enhancement	N/A	N/A	N/A	Active current reducers, feedback network for standing currents.	N/A
Suitability for organic transistors	Not suitable, loading and noise issues.	Potentially suitable, depends on buffer.	Suitable, bandwidth/noise trade-off.	Well-suited if DC reset and noise managed.	Less suitable, low performance, bias noise an issue.

Note: TIA - transimpedance amplifier; VCO - voltage-controlled oscillators; IIR filter - Infinite Impulse Response filter.

Figure S1. S2 LMP7721 and LTC6090 comparison



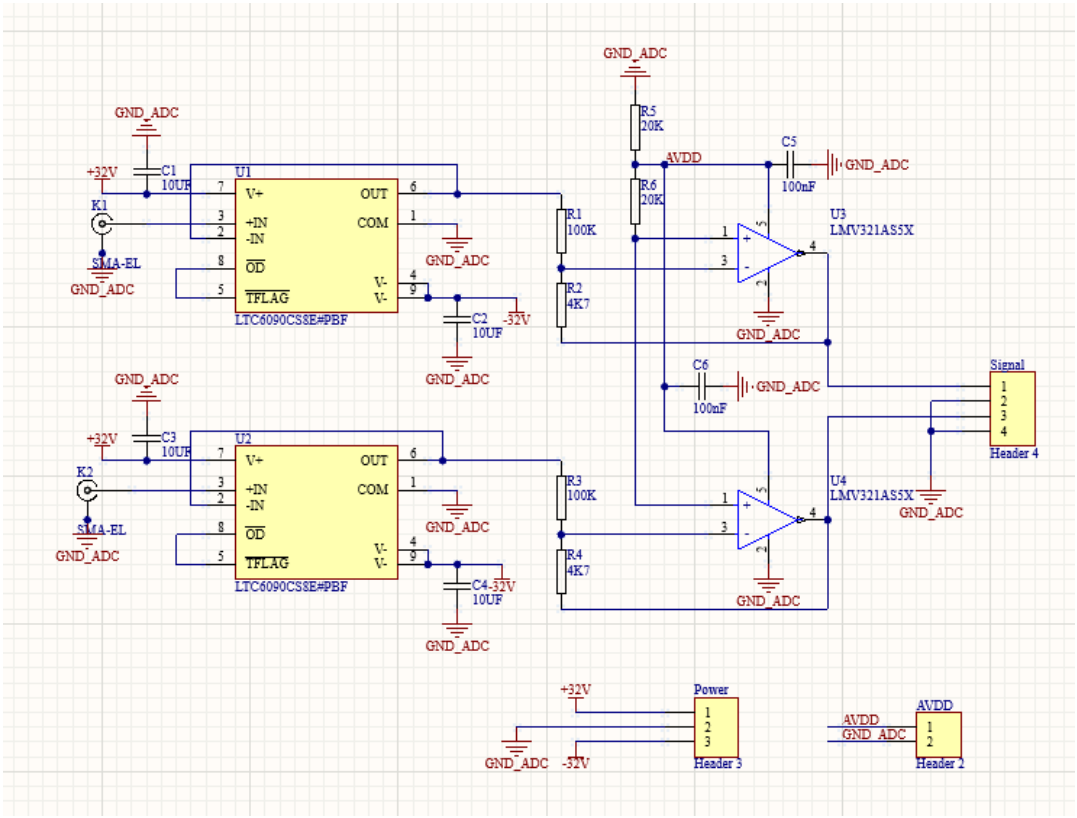


Figure S2. S3 LTSpice model scheme and simulation results

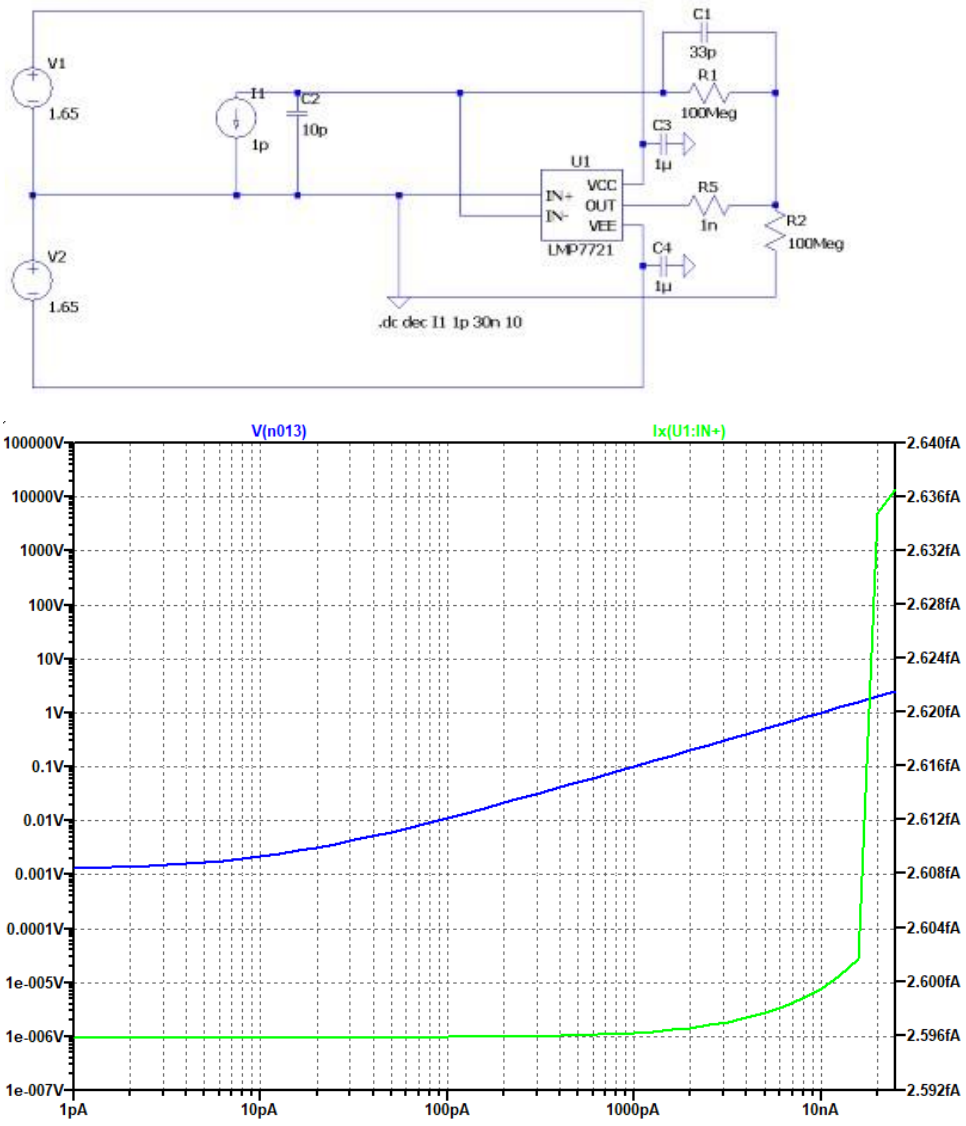


Figure S3. S4 PCB shaping technique

The special technique of shaping the PCB is based on cutouts in the given PCB and revealing the conductive vests that are at the input of the sensitive current meter, thus eliminating the escaping current on the surface of the PCB through passivation, but also minimising the leakage through the volume of the PCB.

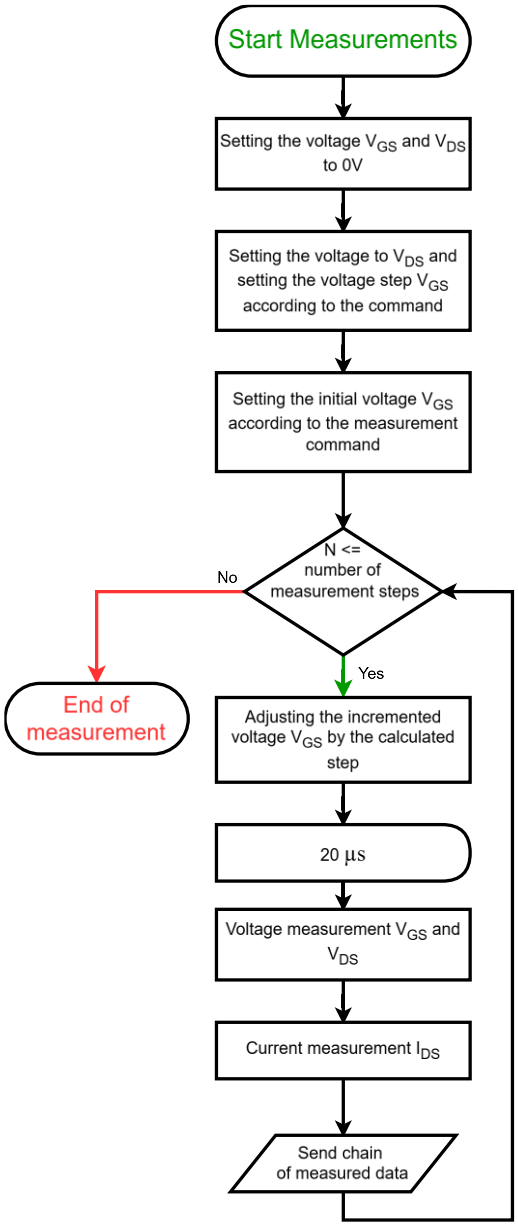


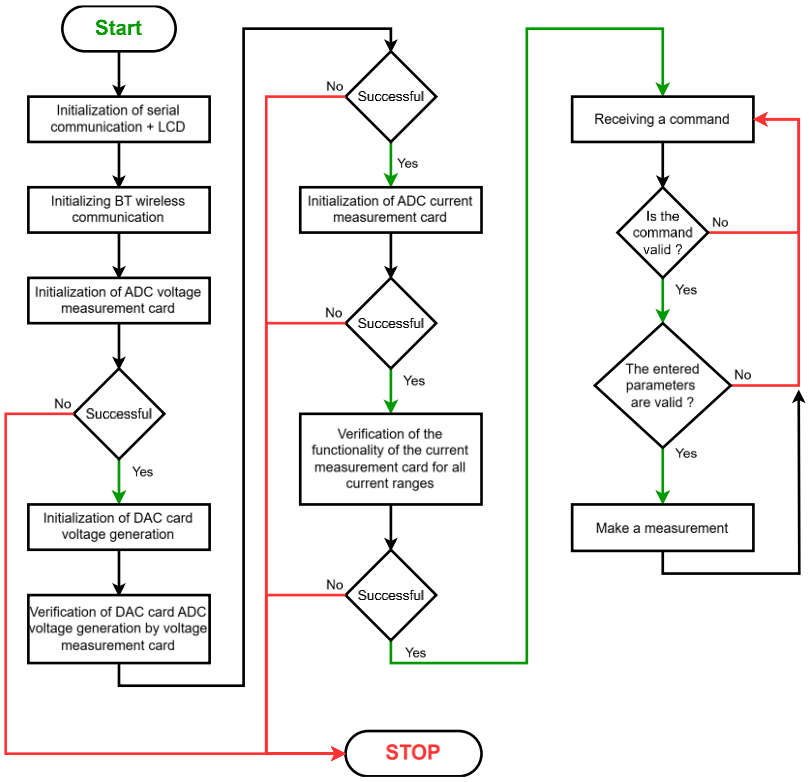
S5 Galvanic separators

The use of galvanic separators to power individual modules was necessary to eliminate ground loops and suppress interference between individual cards. Our specific connection utilizes several TRACO galvanic DC/DC converters. For the current meter's power source, we employ a TRACO TBA 2-0521, which generates +5V and -5V. These voltages are filtered by a 100Hz Pi low-pass filter and subsequently stabilized by low-noise, high-PSSR LDO regulators, the TPS72325 and TLV73325, to +2.5V and -2.5V, respectively, for powering the operational amplifier. The maximum noise of these supply voltages was measured at 3.3 mVrms.

For the voltage measurement and generation source, we use TRACO TBA 2-0521 and TRACO TBA 2-0513 converters. The outputs of these converters are connected in series, yielding voltages of +35V and -35V. These voltages are also filtered by a Pi low-pass filter and then stabilized by low-noise, high-PSSR LDO regulators, the LM317 and LM337, to +32V and -32V, respectively, for powering input and output operational amplifiers. The maximum noise of these supply voltages was measured at 5.1 mVrms.

Figure S4. S6 Measurement flowcharts





S7 Firmware

The self-test is a part of the firmware that can initially test individual cards with each other to see if they work correctly using a defined element connected between individual measuring cards; specifically, it is a control of voltage generation using a voltage measurement card or checking the flowing current with the help of the generated voltage connected through a resistor.

The firmware also initialises serial communication for the display and wireless communication for data collection. If the initialisation of the measurement card fails, then the program halts and displays an error message. Upon successful initialisation, the firmware proceeded with verification measurement on the voltage measurement card. If verification failed, the program halted and displayed the relevant error. If verification was successful, the firmware moved on to proceeded to verify the functionality of the measurement card across all ranges by testing zero in the individual measurement ranges. If verification failed, the program stopped and displayed the respective error.

Once verification was successful, the firmware entered a command reception mode, waiting for a measurement command with parameters. Upon receiving a command, its validity was verified. If the command was invalid, the program returned to the command reception phase, displayed an invalid command message, and waited for a valid command. If the command was valid, the firmware checked the validity of the measurement parameters to ensure they were within the allowed ranges. If the parameters were out of range, the program returned to the command reception phase and displayed an invalid parameter message. If the parameters were correct, the firmware executed the measurement routine described above and returned to command reception upon completion.

S8 Pseudocode

Value name	Type	Description
FIR_const_voltage	Array of floats	Field contains for a 50 Hz notch filter with filter order 15, and a 1 kHz sampling rate
ADC_read_voltage	Array of uint32	The field contains 15 samples of one measured voltage point
ADC_filt_Voltage_A, ADC_filt_Voltage_B,	uint32	The variable contains the filtered ADC value
ADC_read_current	Array of uint32	The array contains 15 samples of one measured current point
ADC_filt_current	uint32	The variable contains the filtered ADC value
Current_range	Char	Contains an indication of the current measuring range
Current_A_H_Zero, Current_A_M_Zero, Current_A_L_Zero	Const uint32	Constants representing zero current in ADC value
Current_A_H_ Inter, Current_A_M_ Inter, Current_A_L_ Inter	Const float	Constants representing the transformation constant of ADC value to current
Measured_current	float	The variable contains the currently measured value
Voltage_A_Zero, Voltage_B_Zero	Const uint32	Constants representing zero volt in ADC value
Voltage_A_inter, Voltage_B_inter	Const float	Constants representing the transformation constant of ADC value to voltage
Measured_voltage_A, Measured_voltage_B	float	The variable contains the currently measured value
Gen_A_Zero, Gen_B_Zero	Const uint32	Constants representing zero volt in DAC value
Gen_A_inter, Gen_B_inter	Const float	Constants representing the transformation constant of voltage to DAC value
Gen_A_DAC, Gen_B_DAC	uint32	The variable contains the DAC value for generating card
Read_CMD	String	Loaded measurement command string
CMD_ind1, CMD_ind2, CMD_ind3, CMD_ind4, CMD_ind5, CMD_ind6,	String	divided values of the measurement command
Vgs_max, Vgs_min, Vds_max, Vds_min,	float	converted command values to float and assigned for individual variables
Vgs_samples, Vds_samples	uint16	converted command uint16 to float and assigned for individual variables
Sweep_step_Vgs, Sweep_step_Vds	float	The variable contains the size of the linear step change for the specified number of samples
Data_out	String	Processed string of measured data

// **Initialization**

Initialize Serial communication

Initialize Bluethoot

Initialize SPI communication

Initialize voltage_ADC (Analog-to-Digital Converter).

if initialization Failed:

call the method Error_Disp with an argument 1 ("voltage measure card error").

Initialize DAC (Digital-to-Analog Converter).

call the method verif_DAC

if verifivation return error:

call the method Error_Disp with an argument 2 ("voltage generator card error").

Initialize current_ADC (Analog-to-Digital Converter)

if initialization Failed.

call the method Error_Disp with an argument 3 ("current card error").

Verifivation of current_ADC zero value in all ranges.

if verifivation Failed:

call the method Error_Disp with an argument 4 ("current ranges error").

// **Main loop**

while (true) {

Wait for a command

If a command is received:

call the method parse_CMD with received argument(Read_CMD).

if command is not valid

restart loop and wait for new command

call the method verif_CMD with parced argument(Vgs_max, Vgs_min, Vds_max, Vds_min, Vgs_samples, Vds_samples).

if command parameters is valid

call the method Measur with an argument (Vgs_max, Vgs_min, Vgs_samples, Vds_max, Vds_min, Vds_samples)

else restart loop and wait for new command

}

// **Functions**

Error_Disp(int "index of error")

sedn to displej message with letter E and index of error.

Stop program in infinitu loop

//

```
verif_DAC()
        LOOP FOR i FROM 1 TO 5
                call the method set_voltage with an argument (i*15-45,1)
                call the method set_voltage with an argument (i*15-45,2)
                call the method read_voltage with an argument (1) and compare withvalues (i*15-45)+-0.01
                if comparison is false
                        return Error
                call the method read_voltage with an argument (2) and compare withvalues (i*15-45)+-0.01
                if comparison is false
                        return Error
        END LOOP
set_voltage(voltage,channel)
        if channel is 1
                set Gen_A_DAC to ROUND(voltage * Gen_A_inter)+ Gen_A_Zero
                send SPI command to DAC with Gen_A_DAC
        els eif channel is 2
                set Gen_A_DAC to ROUND(voltage * Gen_B_inter)+ Gen_B_Zero
                send SPI command to DAC with Gen_B_DAC
////////////////////////////////////////////////
read_voltage(chanel)
        if channel is 1
                send SPI command to change chanel to 1
                LOOP FOR i FROM 1 TO 15
                        send SPI command to start ADC conversion
                        set ADC_read_voltage in index i with read value from ADC via SPI
                END LOOP
                set ADC_filt_Voltage_A with a ROUND mathematical multiplication of ADC_read_voltage and
FIR_const_voltage and subsequent summing of all results
                set Measured_voltage_A to (ADC_filt_Voltage_A - Voltage_A_Zero) * Gen_A_inter
                return Measured_voltage_A
        els eif channel is 2
                send SPI command to change chanel to 2
                LOOP FOR i FROM 1 TO 15
                        send SPI command to start ADC conversion
                        set ADC_read_voltage in index i with read value from ADC via SPI
                END LOOP
```

set ADC_filt_Voltage_B with a ROUND mathematical multiplication of ADC_read_voltage and FIR_const_voltage and subsequent summing of all results

set Measured_voltage_B to (ADC_filt_Voltage_B - Voltage_B_Zero) * Gen_B_inter

return Measured_voltage_B

//

read_current(Current_range)

send SPI command to change chanel to 1

set current measurment range to Current_range

LOOP FOR i FROM 1 TO 15

send SPI command to start ADC conversion

set ADC_read_current in index i with read value from ADC via SPI

END LOOP

if Current_range is "H"

if mean value of ADC_read_current is in range Current_A_H_Zero - 2^4 and Current_A_H_Zero + 2^4

set current measurment range to M

set Current_range to M

call the method read_voltage with an argument (M)

set ADC_filt_current with a ROUND mathematical multiplication of ADC_read_current and FIR_const_voltage and subsequent summing of all results

set Measured_current to (ADC_filt_current - Current_A_H_Zero) * Current_A_H_ Inter

return Measured_current

else if Current_range is "M"

if mean value of ADC_read_current is in range Current_A_M_Zero - 2^4 and Current_A_M_Zero + 2^4

set current measurment range to L

set Current_range to L

call the method read_voltage with an argument (L)

if mean value of ADC_read_current is greater than 2^24-2^4 or smaller than 2^4

set current measurment range to H

set Current_range to H

call the method read_voltage with an argument (H)

set ADC_filt_current with a ROUND mathematical multiplication of ADC_read_current and FIR_const_voltage and subsequent summing of all results

set Measured_current to (ADC_filt_current - Current_A_M_Zero) * Current_A_M_ Inter

return Measured_current

else if Current_range is "L"

if mean value of ADC_read_current is greater than 2^24-2^4 or smaller than 2^4

set current measurment range to M

```
                    set Current_range to M
                    call the method read_voltage with an argument (M)

                set ADC_filt_current with a ROUND mathematical multiplication of ADC_read_current and FIR_const_voltage and subsequent summing of all results
                set Measured_current to (ADC_filt_current - Current_A_L_Zero) * Current_A_L_ Inter
                return Measured_current
////////////////////////////////////////////////
parse_CMD(Read_CMD)
        set CMD_ind1 as 1 pased value from Read_CMD
        set CMD_ind2 as 2 pased value from Read_CMD
        set CMD_ind3 as 3 pased value from Read_CMD
        set CMD_ind4 as 4 pased value from Read_CMD
        set CMD_ind5 as 5 pased value from Read_CMD
        set CMD_ind6 as 5 pased value from Read_CMD
        set Vgs_max as float value from strig CMD_ind1
        set Vgs_min as float value from strig CMD_ind2
        set Vds_max as float value from strig CMD_ind3
        set Vds_min as float value from strig CMD_ind4
        set Vgs_samples as uint16 value from strig CMD_ind5
        set Vds_samples as uint16 value from strig CMD_ind6
////////////////////////////////////////////////
method verif_CMD(Vgs_max, Vgs_min, Vds_max, Vds_min, Vgs_samples, Vds_samples)
        if Vgs_max is smaller than Vgs_min or it is greater than number 32
                return nonvalid -1
        if Vgs_min is greater than Vgs_max or it is smaller than -32
                return nonvalid -1
        if Vgs_samples is smaller than 1
                return nonvalid -1
        if Vds_samples is smaller than 1
                return nonvalid -1
////////////////////////////////////////////////
Measur(Vgs_max, Vgs_min, Vgs_samples, Vds_max, Vds_min, Vds_samples)
        call the method set_voltage with an argument (0,1)
        call the method set_voltage with an argument (0,2)
        set Current_range to 'H'
        LOOP FOR j FROM 1 TO Vds_samples
```

```
        call the method set_voltage with an argument (Vds_min+((j-1)*Sweep_step_Vds),1)
        LOOP FOR k FROM 1 TO Vgs_samples
                call the method set_voltage with an argument (Vgs_min+((j-1)*Sweep_step_Vgs),2)
                delay 20us
                set Measured_voltage_A from method read_voltage with an argument(1)
                set Measured_voltage_B from method read_voltage with an argument(2)
                set Measured_current from method read_current with an argument(Current_range)
                set Data_out string with data Measured_current, Measured_voltage_A, Measured_voltage_B
                Send Data_out via Bluethoot
        END LOOP
END LOOP
call the method set_voltage with an argument (0,1)
call the method set_voltage with an argument (0,2)
```