Technology by INNOVATION LAB thinking works

Software for Printed Sensor Matrices

SensorMatrixLAB

User manual

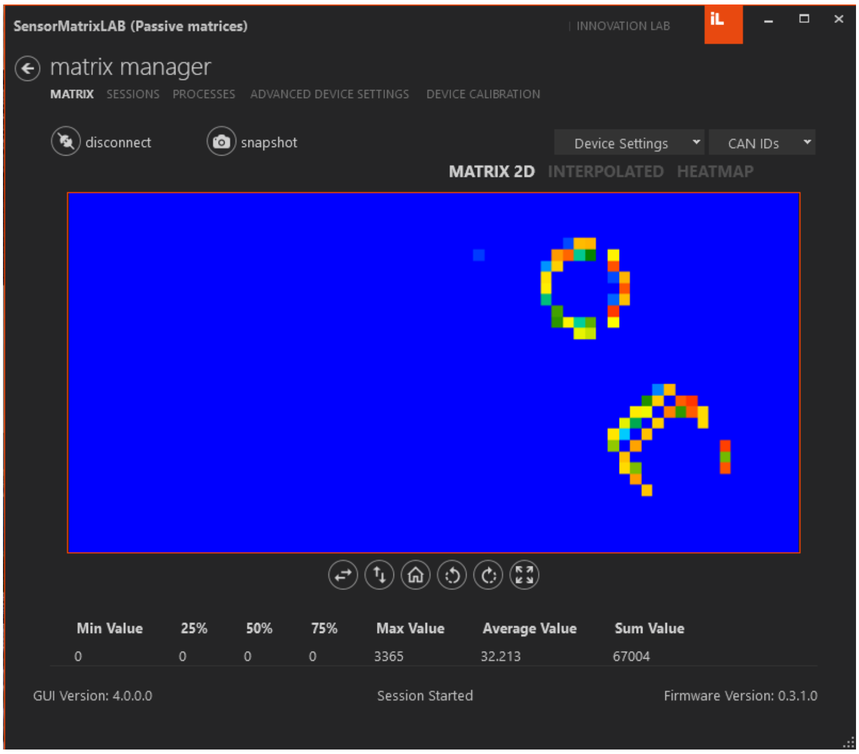
Table of Contents

Table of Contents 2
General information 3
Hardware requirements 5
Software requirements 5
Installing the Software 6
Activating the Software 7
Updating the Software 8
The user interface of the SensorMatrixLAB software 9
Main User Interface Elements 9
Settings 11
Menu “Connection” 11
Menu “Frame Processing” 12
Menu “Hardware type” 14
Menu “View” 14
Menu “Storage” 15
Menu “API Access” 17
Menu “Appearance” 20
Device Settings 21
CAN Settings 24
Tabs 25
Matrix 25
Sessions 26
Processes 27
Advanced Device Settings 28
Device Calibration 33
Software Revision 38
Appendix A. Measurement principle 39
Appendix B. Frame processing with config.json file 40

General information

The “SensorMatrixLAB” software is a Microsoft Windows application developed to visualize and process the data from InnovationLab’s Pressure Sensor Matrixes. The software supports following main features:

- Advanced data visualization,
- High speed data recording and replaying,
- API sensor data access,
- Communication interfaces: USB Serial, Wi-Fi, Ethernet,
- Support of customized printed sensor matrixes and different hardware types,
- Advanced hardware configuration,
- Configuration of readout electronics, including standalone operation via CAN / RS232,
- Sensor calibration (including individual calibration of sensors in the matrix)
- Full proprietary protocol support.



Main window of SensorMatrixLAB software

Contact information

The correspondence related to the current manual as well as any questions related to the software should be addressed to:

InnovationLab GmbH
Speyerer Straße 4
69115 Heidelberg, Germany

T +49 6221 - 54 19 100
F +49 6221 - 54 19 110
E software@innovationlab.de

Hardware requirements

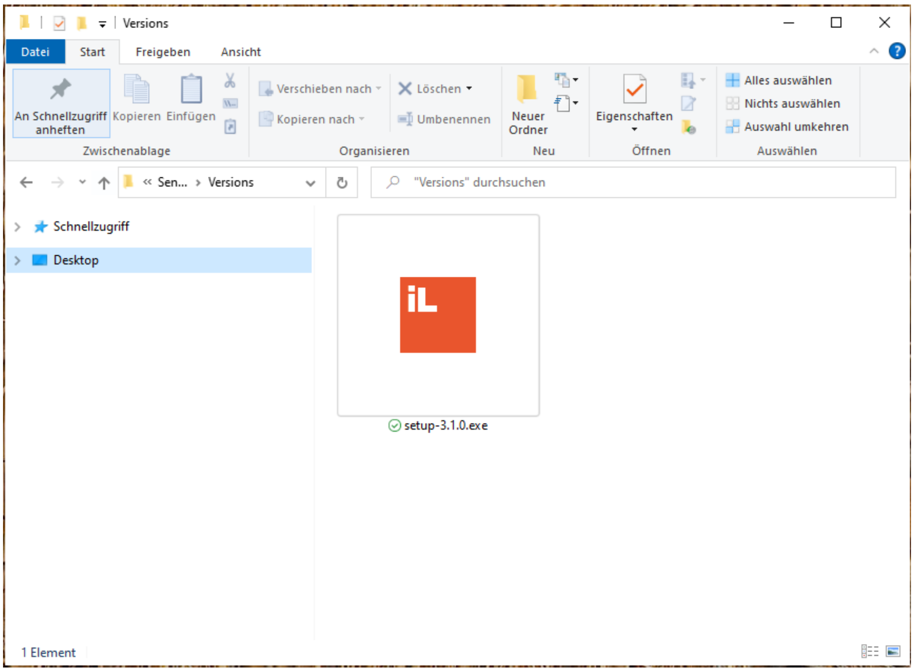
- InnovationLab's hardware (electronics) with a latest firmware version
- InnovationLab's printed sensor matrix
- Personal computer (Windows-based) with x86/x64 processor
- USB port, USB-to-serial driver installed
- Internet connection
- Optional: Connection to local network via Ethernet (for ETH connectivity)

Software requirements

- Microsoft Windows 10 or higher
- Microsoft .NET framework 4.7.2
- Web browser (Mozilla Firefox, Google Chrome, or Microsoft Edge) of latest version
- Microsoft Excel (2016 or higher)

Installing the Software

To install a software on your PC running Windows 10 or higher you should run the received executable installer file:

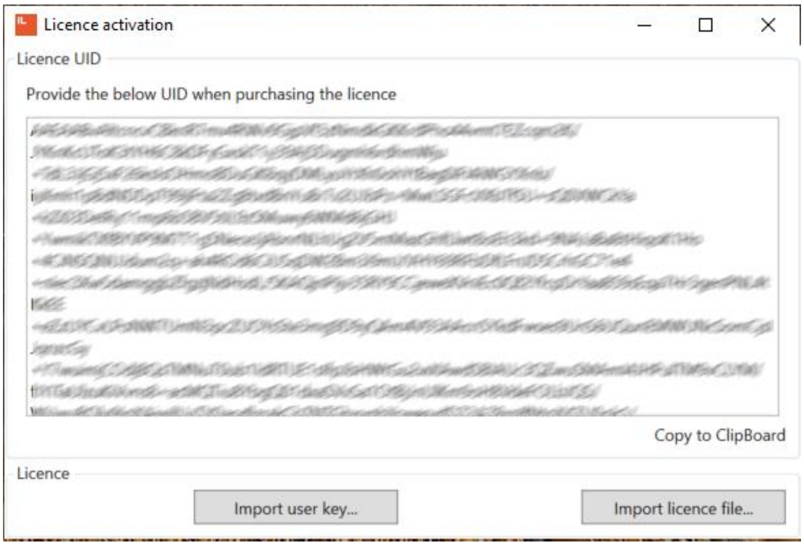


*Locating the Installer file *.exe in Microsoft Windows*

Note that SensorMatrixLAB does not require a local user to have administrator privileges on the machine to install or use the software. However, administrator privileges might be needed to use auto-discovery function.

Activating the Software

After installation the software should be activated. To do so, run the software using a shortcut. You will be prompted to activate.

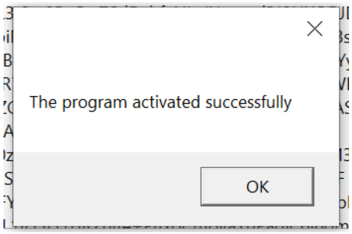


Activation Window

Copy your *License UID* into the clipboard and request a license via InnovationLab's web site. For the activation you will get two files:

- User key
- License file

To activate, **import the user key** in activator, then **import the license file.** If all operations were done correctly, you will see the confirmation:



Congratulations 👏 the license activation is done!

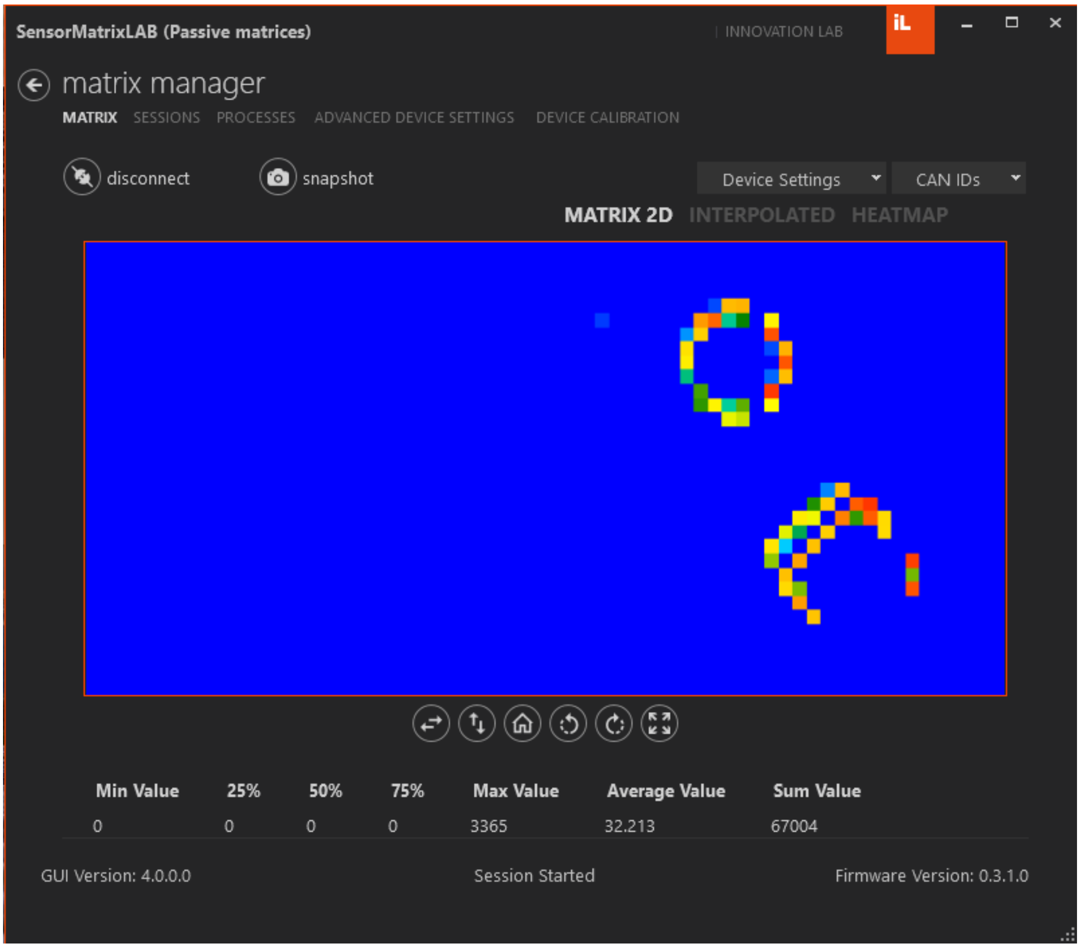
Updating the Software

In order to update SensorMatrixLAB to a newer version follow these steps:

1. Uninstall current version from your PC
2. Download a latest version from InnovationLab’s web site
3. Install new version and perform activation same as for first installation.

The user interface of the SensorMatrixLAB software

Figure below shows the main window of the “SensorMatrixLAB” software with its main elements:



SensorMatrixLAB software - session started

Main User Interface Elements

Settings - global settings of an application:

- Menu “Connection”
- Menu “Frame Processing”

- Menu “View”
- Menu “Hardware type”
- Menu “Storage”
- Menu “API Access”
- Menu “Appearance”

Device Settings

Session settings of the hardware

CAN Settings

Settings for Controller Area Network (CAN bus)

Tabs

- Matrix: represents a color-coded 2-dimentional view of sensor data
- Sessions: contains previously captured sessions
- Processes: information about background threads related to Software
- Advanced Device Settings: time-domain filtering, Wi-Fi, In-board frame processing
- Device Calibration: enables sensor calibration (each element in the matrix individually)

Content View

Element is used for visual demonstration of the live sensor data or recorded sessions

App and Firmware version

Version of the application and firmware

Connect / disconnect

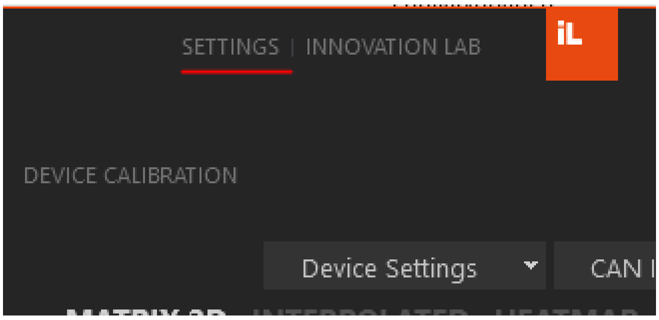
Button for starting and stopping a session

Frame statistics

Current sensor frame information

Settings

Global settings concerning an application and connection to hardware can be changed under ***SETTINGS***, top right corner of main window.



SensorMatrixLAB software - Settings

General application settings are saved in local user folder. Categories of settings are listed in sub-sections below.

Menu "Connection"

The *CONNECTION* section allows to select between connection interfaces:

- USB connection
- TCP/IP

Not all connection methods are supported by each hardware. Refer to hardware manual for more information.

The **USB connection** is used as a virtual serial port. To use it, a proper driver installation is required on your Windows machine. Contact your system administrator to install proper drivers.

In case of USB connection, the proper port should be selected. Refer to *Windows Device Manager*, section *Ports* for available devices.

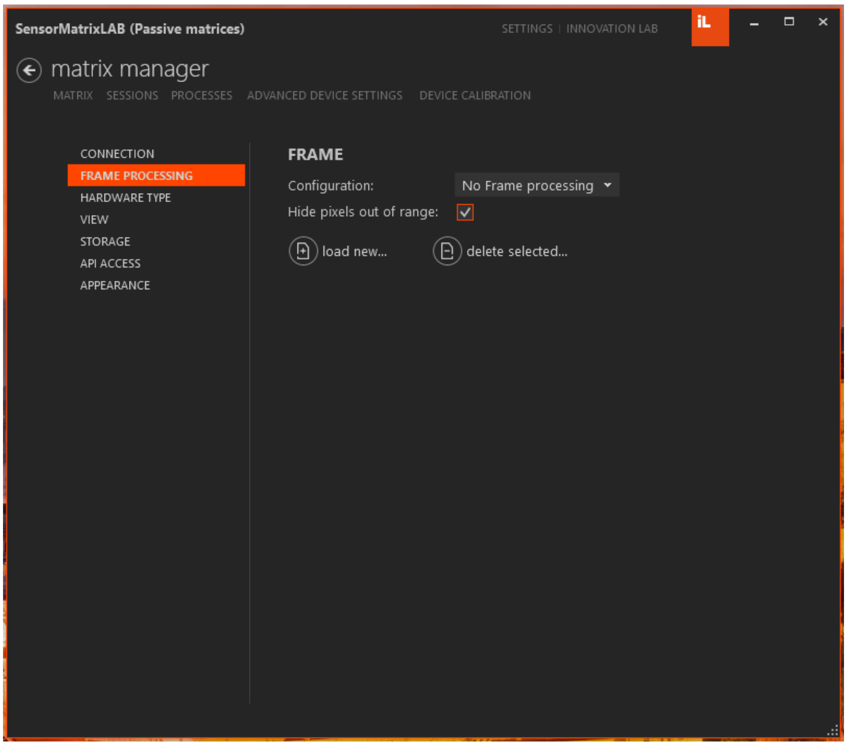
The USB baud rate should correspond to used hardware. Refer to hardware manual for this value. For most InnovationLab's boards `baudrate = 1000000`.

The **TCP/IP connection** enables data transfer from the board with Ethernet or Wi-Fi. For correct operation the hardware should be in the same network, as the Windows machine. At the same time, correct hardware *IP address* and *port* should be specified. Use **Scan** button for automatically acquiring this information. *IP address* and *Port* of the *Hardware* can be also found in your router in connected devices list, and then input manually. If you need an information regarding hardware *IP address* and *Port*, contact your system administrator.

Note that for using the Wi-Fi connection, the prior hardware configuration is necessary. Refer to section **Advanced Device Settings** for more details.

Menu "Frame Processing"

The *Frame Processing* section allows configuring the frame representation from sensor data, coming from the board.



Settings – menu "Frame Processing"

Configuration - corresponds to the type of Printed Sensor Matrix attached to the hardware. The selection of this fields will impact how the data stream coming from your Hardware will be converted into a 2D visual representation in the Software. The

PC rearranges the byte order for proper visualization. This option is implemented to support different types of sensor matrixes on the same hardware. The selected “Configuration” should match with the sensor type used.

The list of available Configurations is determined by the content of `frame.config` file, located in:

`C:\Users\{user}\AppData\Roaming\INNOVATION LAB\SensorMatrixLAB\.`

If option “**No Frame Processing**“ is selected in Frame/Configuration, the data stream will not be processed on the PC side and the frame will be visualized as received from the board. The matrix size for visualization will be taken as Width and Height as specified in Device Configuration.

Use “**No Frame Processing**“ option in case that raw sensor data without rearrangement is needed or in-board rearrangement is used.

If the proper Configuration for Printed Sensor Matrix cannot be found, contact InnovationLab GmbH. Alternatively, build your own `frame.config` file and add it as described below. Refer to **Appendix B. Frame processing with config.*json*** file for more information.

Note that **Hardware-side rearrangement** and **PC-side rearrangement** can interfere each other. Use only one type of rearrangement at the same time.

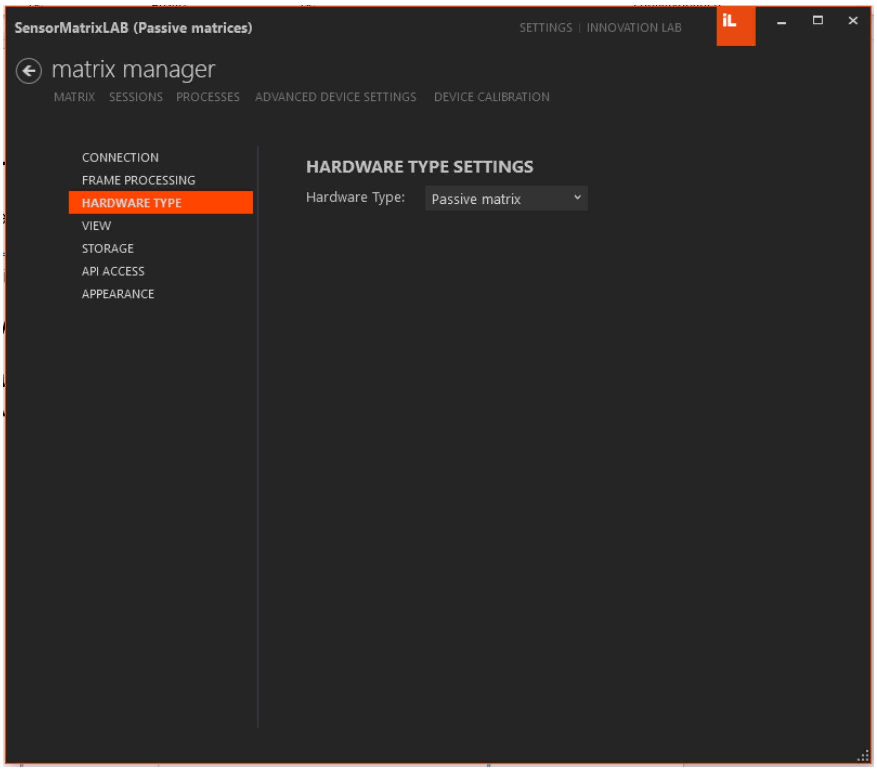
User can add new configurations into the SensorMatrixLAB software to support new matrix types. To do so, use the **Load new** button. Using load file dialog, navigate to your *.json file, containing Frame configuration received from InnovationLab. Alternatively it is possible to prepare the *.json Frame configuration according to **Appendix B. Frame processing with config**.json file. If the loaded file is correct, the form will ask which of the found configurations should be added. If configuration with the same name already exist, a warning will be shown. After configuration is loaded, select it in the list to apply.

When needed, the configuration can be deleted with **Delete selected** button.

The checkbox **Hide pixels out of range** will enable the software to hide from the heatmap view all data pixels, which have a failure flag in the incoming data stream. This flag is set on in the hardware when a certain sensor data is acquired, but might be incorrect, e.g. out of range measurement.

Menu "Hardware type"

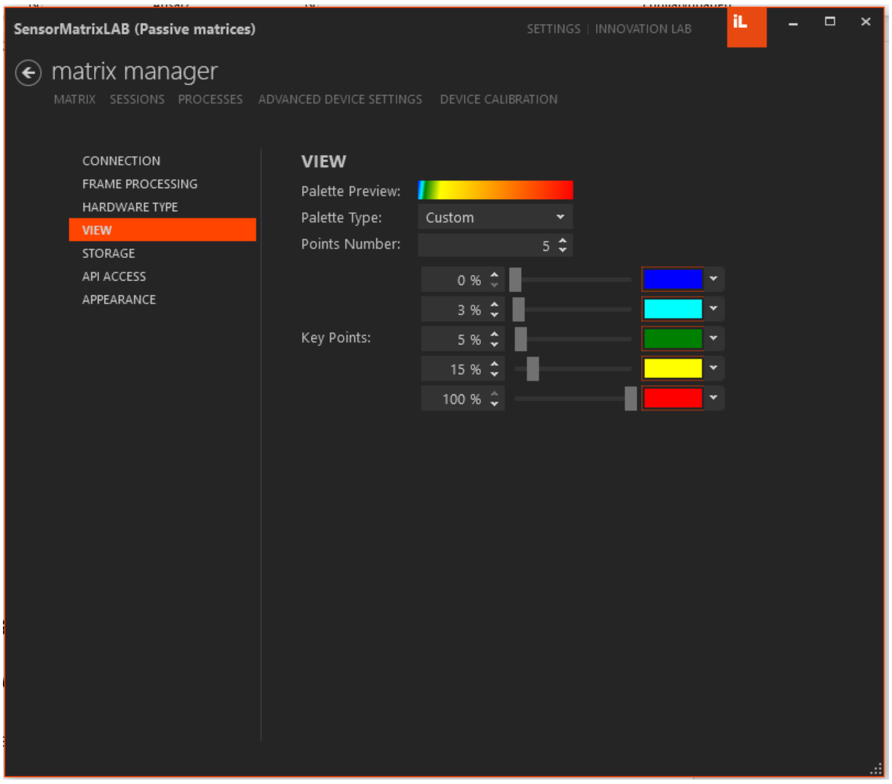
This page enables switching SensorMatrixLAB to work with non-standard hardware types, e.g. ActiveMatrix. Note, that not every license of SensorMatrixLAB supports this functionality.



Settings – menu "Hardware Type"

Menu "View"

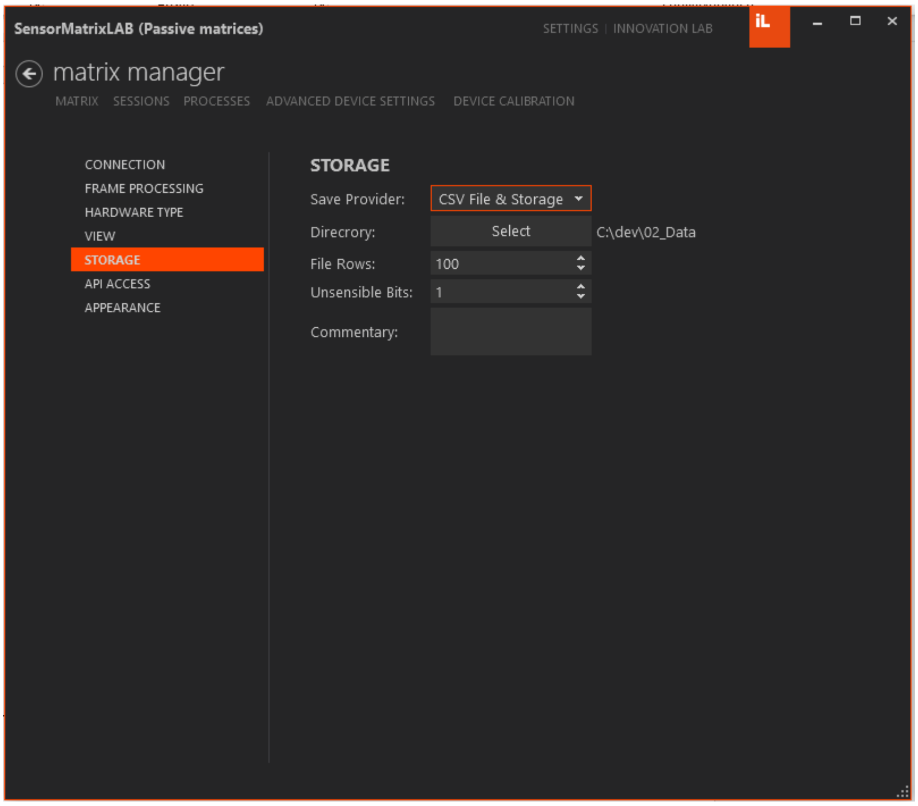
The *VIEW* section allows to select and configure a color map for data presentation in the main window.



Settings – menu "View"

Menu "Storage"

The *STORAGE* section allows saving of a session in two ways: *CSV* (csv file) or *Database* (storage file). Selection of these modes can be made under **Save Provider.** Make sure that a **Directory** is set. Both data types will be saved in this path.



Settings – menu “Storage”

To prevent unnecessary large **.csv* files, it is possible to split a session into several files. **File Rows** defines the maximum number of rows in a single **.csv* file. One row is presenting a single frame. If this amount is reached, a new **.csv* file will be created.

It is also possible to reduce data size using the **Unsensible Bits** option. This value represents a range where the sensor values can vary without being saved. This can be used for long time measurements with not much action and will prevent that insignificant noise will be saved.

The data in **.csv* files is arranged in following format:

Column	1	2	3	4 and following
Description	Time Stamp (in ms)	Sample Number	CAN Timestamp (if configured)	Data of sensor pixel

A text field **Commentary** can be used to add a text string to the captured data.

Additionally to data itself, a *session.txt* or *session.db* file will be generated, containing important information about session. An example file can be seen below (Listing 1).

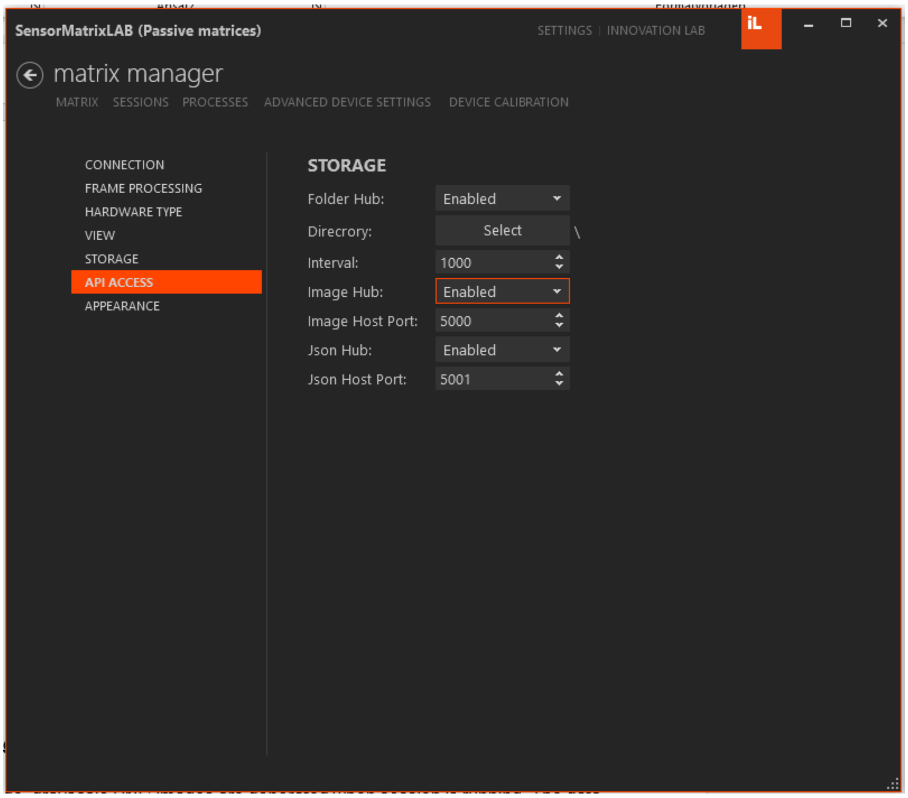
Listing 1 – Session description file session.txt

```
SESSION
Start Time:             2021-06-07 13:12:28
Stop Time:              2021-06-07 13:12:33
Session Time:           5,18 seconds
Received Samples:       97
Saved Samples:          97
Frequency:              18,72 Hz


SETTINGS
Frame Configuration:    65x20pix_530x105mm
Frequency:              unlimited
Switch-read Delay:      0 us
Number of Samples:      unlimited
CAN Timestamp:          1970-01-01 00:00:00
Firmware version:       0.2.6.9
GUI version:            3.0.1.0
Unsensible Bits:        2
Commentary:
```

Menu “API Access”

The **API Access** option enables live data integration from InnovationLab’s software into other programs on the same PC. Integration options available are described below.



Settings – menu "API Access"

PNG-Image, interval-based generation

In this mode, grayscale PNG images are generated when session is running. The data is written into a selected **Directory** with a selected **Interval** (in ms). To turn on this mode, use **Folder Hub** setting.

Data generated in this mode can be viewed with any Windows based graphics viewer/editor, e.g. MS Paint. Each pixel of the created graphics is corresponding to the pixel of sensor matrix, and the luminosity contains information about sensor value.

The following configuration is used for PNG encoding (**Listing 2**):

Listing 2 – configuration of PNG encoding:

```
ColorType = Grayscale;
BitDepth = PngBitDepth.Bit16;
CompressionLevel = PngCompressionLevel.NoCompression;
FilterMethod = PngFilterMethod.None;
InterlaceMethod = PngInterlaceMode.None;
```

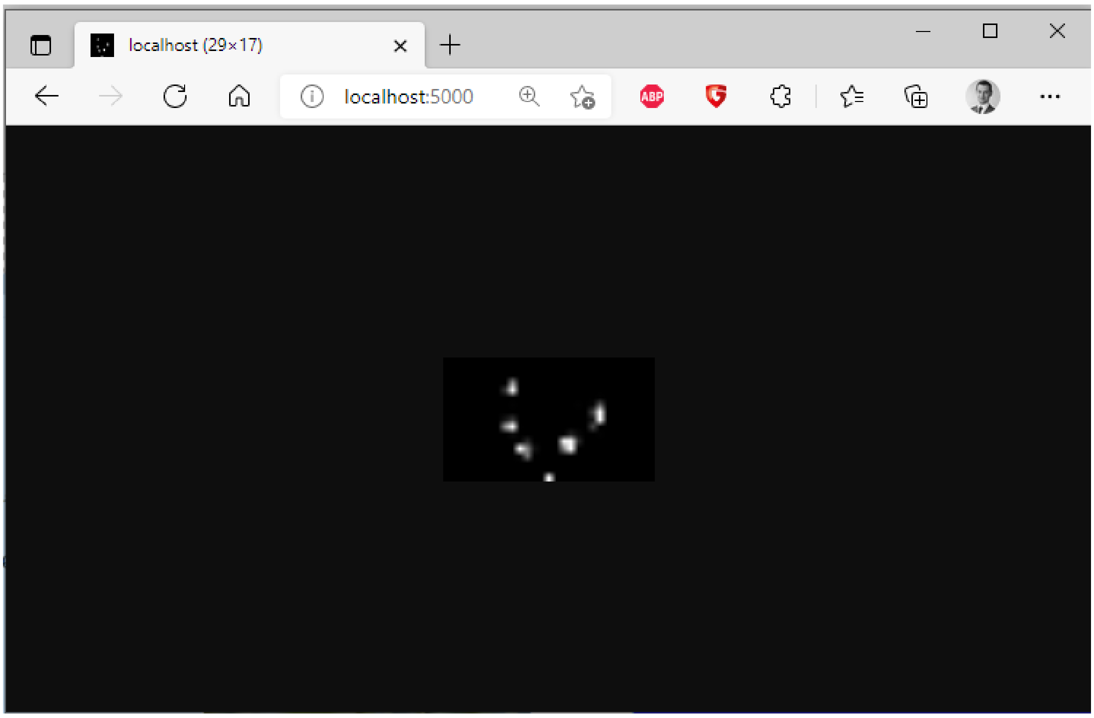
To ensure the high contrast in graphics viewer program, the 12-bit sensor data in 16-bit PNG is aligned left. This means, that 12 most significant bits of 16-bit luminosity values are used and remaining 4 bits can be ignored. To get sensor raw values from saved PNGs, luminosity values be shifted 4 bits right (**Listing 3**):

Listing 3 – example PNG decoding in C# using *SixLabors.ImageSharp library*

```
for (var i = 0; i < width; i++)
 {
     for (var j = 0; j < height; j++)
     {
         image[i, j] = new L16((ushort)(PNG_value[i, j] >> 4));
     }
 }
```

PNG-Image, API access

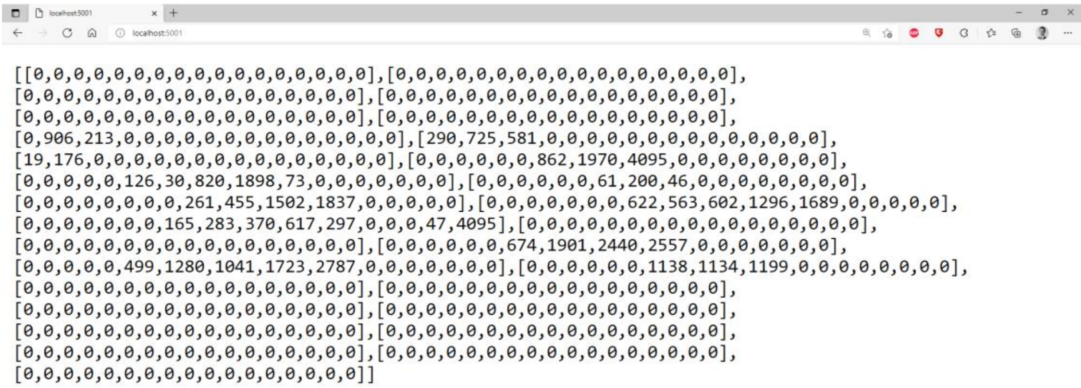
In this mode, the matrix state is generated as a *.png image. This happens on demand upon incoming HTTP requests. Images can be loaded using the HTTP GET request via the selected **Image Host Port**. The data format is the same, as in interval-based generation. To turn on this mode, use **Image Hub** setting.



Example PNG file containing matrix data loaded via Web browser from localhost:5000

JSON file, API access

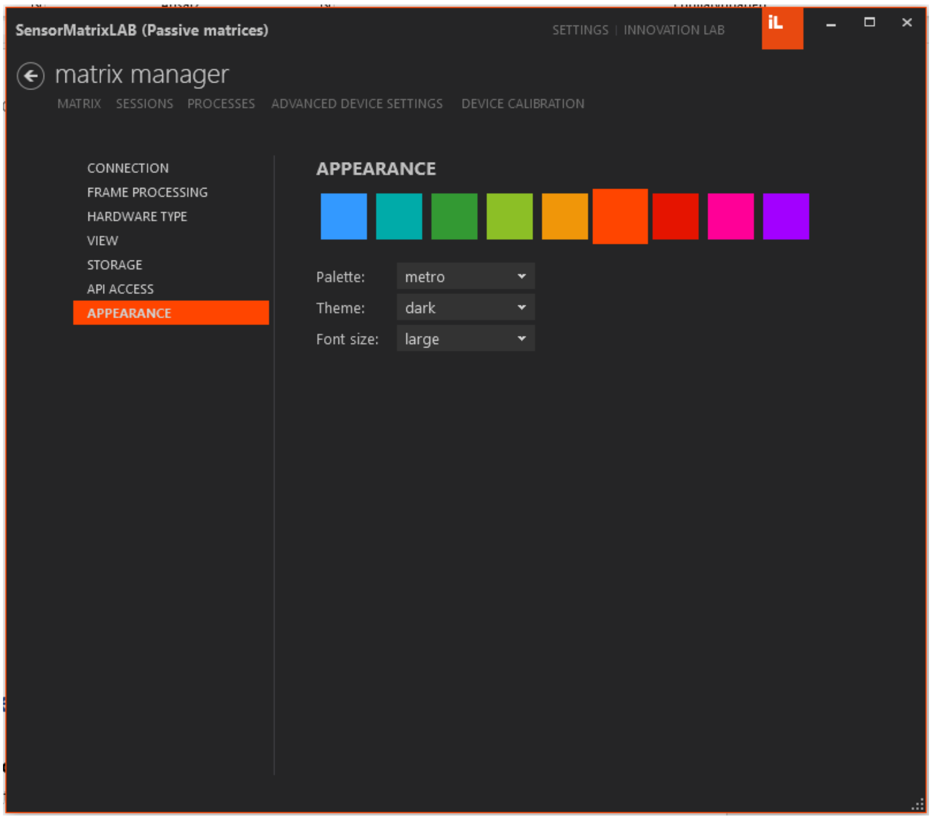
In this mode, the matrix state is generated as JSON file. This happens on demand upon incoming HTTP requests. The data can be loaded from a third-party software using the HTTP GET request via the selected **JSON Host Port**. To turn on this mode, use **JSON Hub** setting.



Example JSON file containing matrix data loaded via Web browser from localhost:5001

Menu "Appearance"

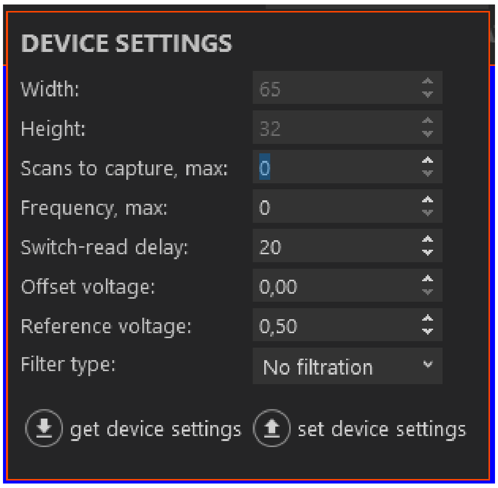
All settings about the app style can be found in the **APPEARANCE** section.



Settings – menu "Appearance"

Device Settings

The **Device settings** are saved in the non-volatile memory of the hardware and the same settings will be used for controlling the device via all supported interfaces (e.g. CAN, ETH). Note that to update the device configuration, an existing connection to the hardware is required and session must be stopped.



Device Settings

Providing that the USB connection to the PCB is successfully established, PCB settings can be changed in main window under *Device Settings.* The **get device settings** button loads settings from the board, **set device settings** sends user's input to the board.

- **Width** and **Height** describe the scan area size that will be used during the session. These values are *read-only*. Width and Height of the scan are corresponding to the maximum resolution of the hardware in case in-board frame converter is *off* or to actual selected frame size in case in-board frame converter is *on*. Refer to hardware datasheet for more information.
- **Scans to capture, max** makes it possible to limit the number of captured frames of each session. After this number is reached, session will be stopped. If this value is set to 0, session is not limited and can only be stopped using *disconnect* button.
- **Frequency, max** describes the upper limit for the measurement frequency of a session in Hz (samples per second). If this value is set to 0, measurement will be done with maximum frequency possible for current hardware and sensor matrix. Note that some hardware does not support frequency limitation. Refer to hardware datasheet for more information.
- **Switch-Read Delay** allows to set a delay between multiplexing and reading each matrix pixel. This value might influence the image quality and provides control over timings. The unit of this value is µs.

- **Voltage Offset** allows to set an offset voltage value. The unit of this value is Volts.
- **Voltage Reference** allows to set a reference voltage value on the board. In general, the higher the reference, the higher is the sensitivity of the sensor system to external stimulus. It is recommended to start measurements with low reference voltage and increase it when needed. The unit of this value is Volts.
- **Filter Type** allows applying time-domain filtering on live sensor data. This means that each matrix pixel is filtered as a function of time, independently from other pixels. Note that not all hardware support live filtering. For more information refer to the manual for corresponding hardware.

Refer to **Software Revision**

Version	Date	Description
3.1.4	15 Sept 2021	Initial public release
3.1.5	22 Feb 2022	• Changes in default UI settings (after installation); • The license is independent of the BIOS version of the PC.
4.0.0	8 March 2022	• Added 32-bit values support into USB / WLAN protocol; • Added non-standard hardware types support; • Sensor calibration added; • Added displaying of current pixel value on mouse hover; • Added displaying up to 16x time-series charts of pixel data; • Frame statistics reworked (added percentiles); • License grades: BASIC / ADV / PRO.

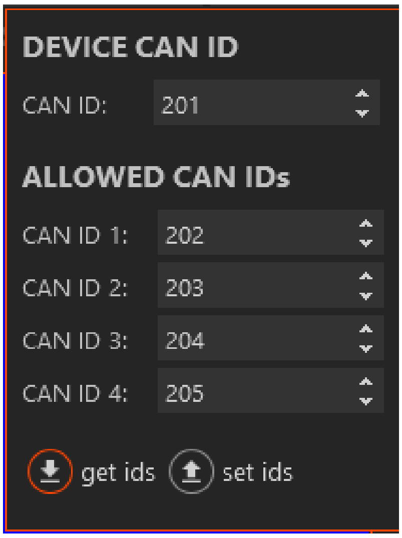
Appendix A. Measurement principle for more information.

- If get/set settings buttons are not functioning, check the connection settings under ***SETTINGS* - *CONNECTION***.
- Download and flash the latest Firmware Version into the electronics.

CAN Settings

Section CAN settings are related to communication over Controller Area Network (CAN bus). In order to get or set these settings in hardware, a working USB / TCP connection is required. Following parameters can be set:

- CAN ID – defines the ID of the hardware. Default value is 201.
- CAN ID x - defines the CAN IDs of devices whose commands will be accepted by the hardware.



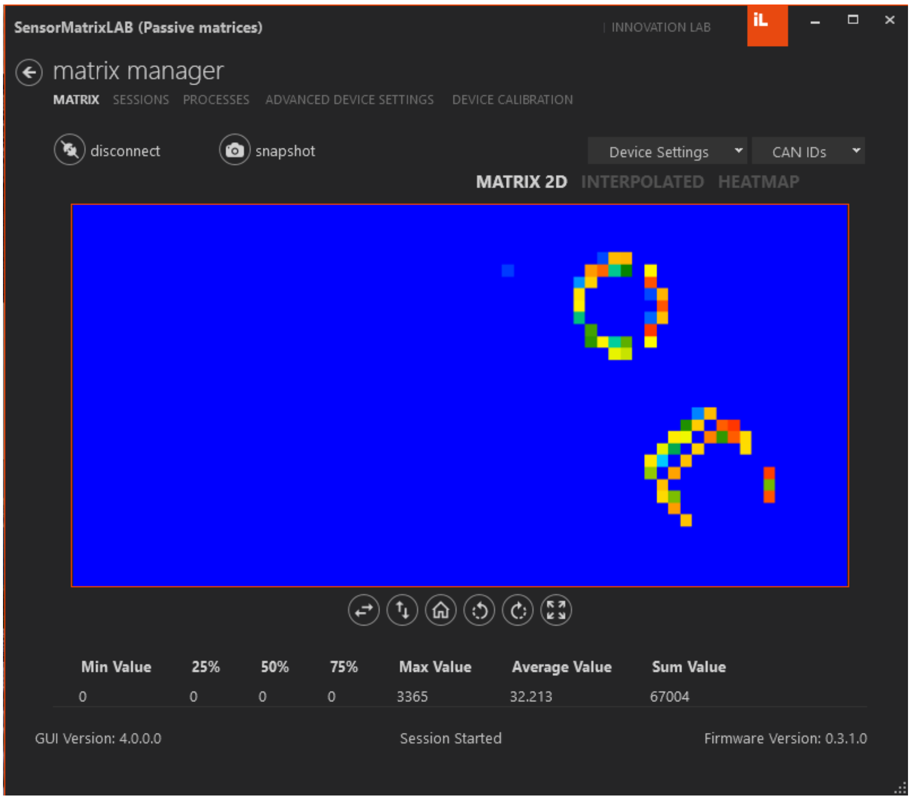
CAN Settings

Tabs

The tabs of main window provide access to the core functionalities of SensorMatrixLAB – visualizing and saving sensor data as well as advanced board configuration. Refer to the sections below for detailed explanations of each tab.

Matrix

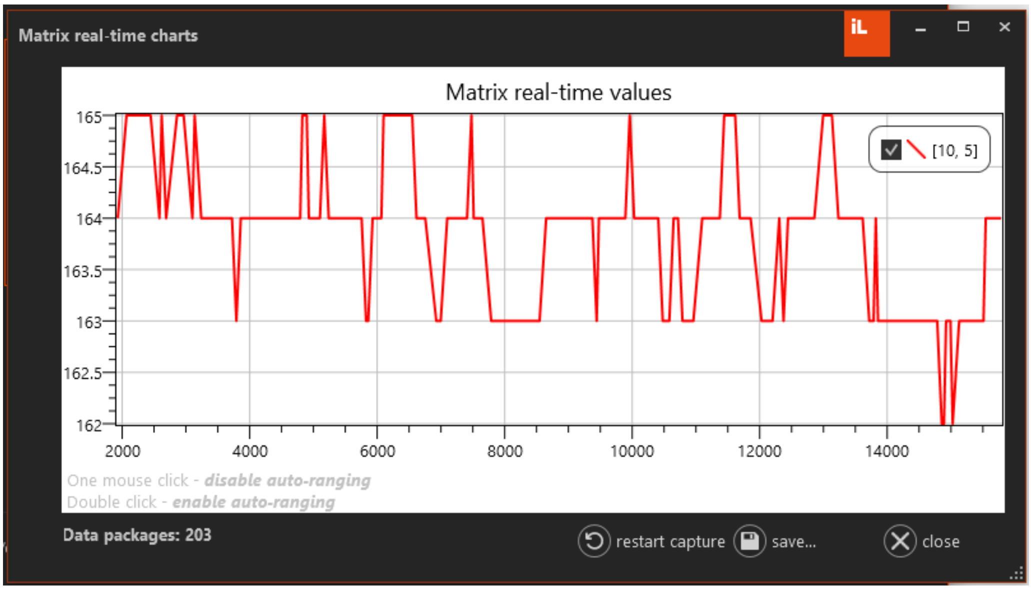
This tab contains live sensor data visualized as a heatmap.



SensorMatrixLAB – tab «Matrix» (2D view)

Additional sensor data representations are available on this tab: 2D raw, interpolated, heatmap.

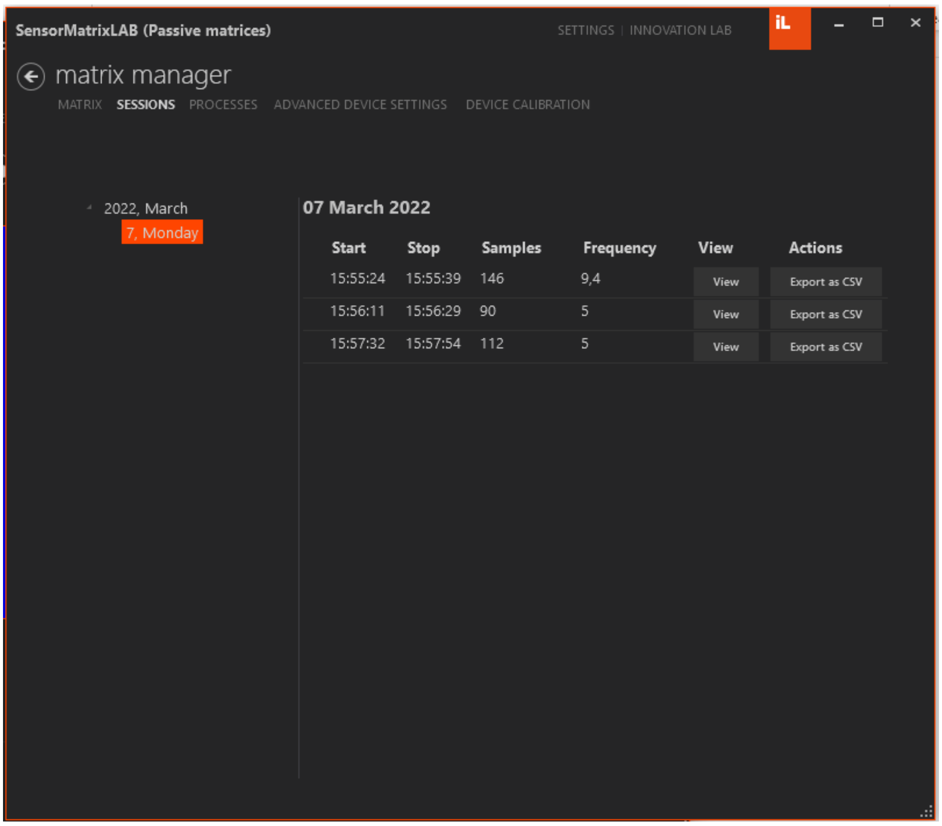
Its is possible to visualize the matrix pixel data as a time series. Corresponding window is opened by doble-click over the target pixel. In this mode up to 16 pixels can be visualized.



Time-series view of individual pixel

Sessions

This tab contains information about previously recorded sessions. Note that only sessions saved in database format are displayed. To select a folder for storing sessions, use **Settings - Storage** menu, field **Directory**.

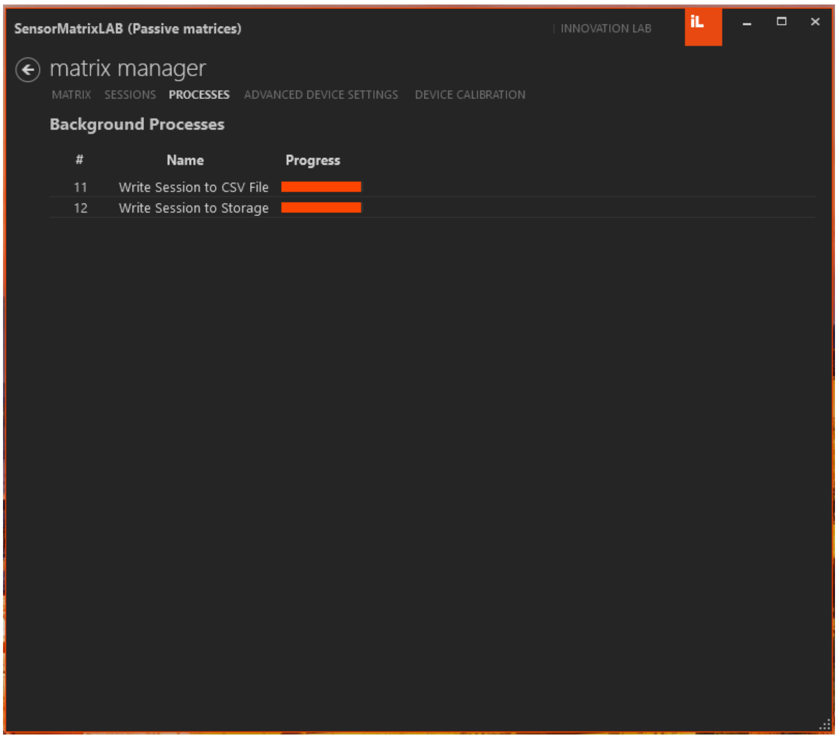


SensorMatrixLAB – tab “Sessions”

From this tab, the user can play previously recorded sessions in an embedded player or export the selected session into CSV format.

Processes

This tab contains a list of subprocesses created by SensorMatrixLAB, such as database writer, CSV writer or other. The **Progress** bar represents the current load of the process. Note that the performance of subprocesses can be affected by the accessibility of saving destination. In this case, please consider using another destination **Directory** in **Settings - Storage** menu.

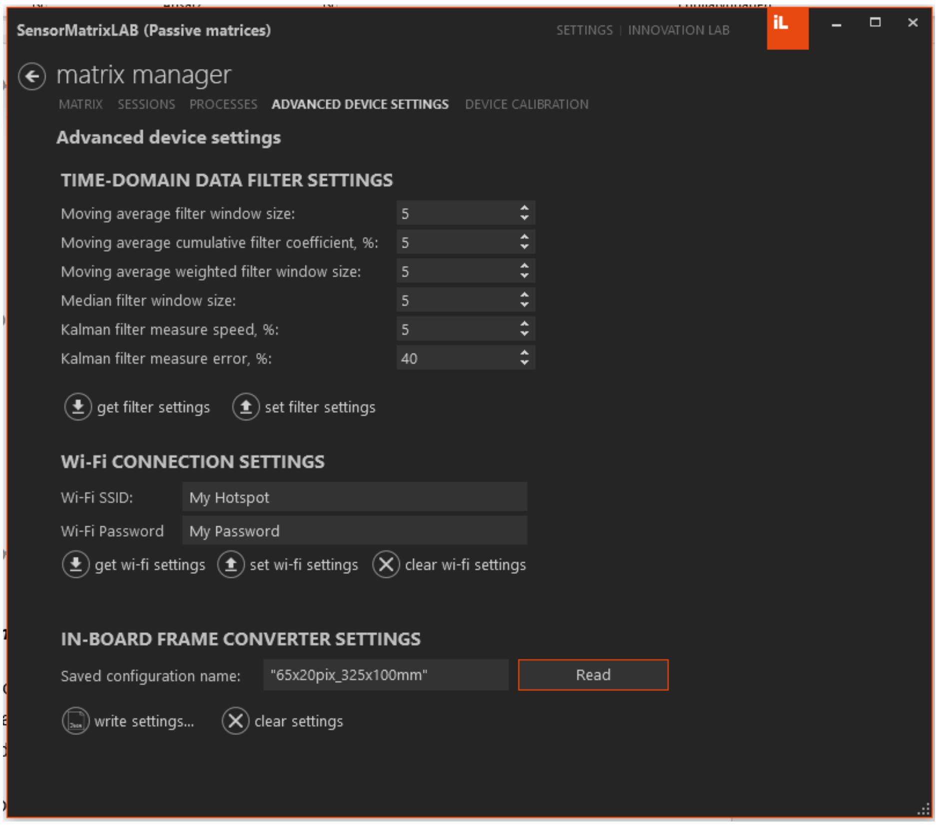


SensorMatrixLAB – tab “Processes”

Advanced Device Settings

The tab **Advanced Device Settings** allows the user to configure advanced in-board functions:

- Hardware-side time-domain filters
- Hardware wi-fi connection
- Hardware-side frame rearrangement



SensorMatrixLAB – tab “Advanced Device Settings”

Time-domain data filter settings

In this section, the user can configure the time-domain filtering of sensor data. This means, each pixel is processed individually without being taking into account the data generated by its neighbors. Below short description of filters is given, more specific information and mathematic explanation can be found on the web.

Following filter algorithms are available in SensorMatrixLAB:

Moving average

In this filter an unweighted mean of the previous *k* data-points is taken as an output. The parameter *k* represents the **window size.**

Exponentially weighted moving average

An exponentially weighted moving average (EWMA) applies on input data-points weighting factors which decrease exponentially. The weighting for each older datum decreases exponentially, never reaching zero. The output is calculated as:

$S_i = Y_0, i = 0,$

$S_i = \alpha Y_i + (1 - \alpha) Y_{t-1}, i > 0,$

where S_t is output of filter with an index i, Y_i is measurement with an index i, α is a **cumulative filter coefficient.**

Moving average weighted

A weighted average is an average that has multiplying factors to give different weights. to data at different positions in the sample window. *k* represents the **window size**, at which the weights decay linearly: In an *k*-size window latest measurement has a weight of *k*, second last *k-1* and so on.

Median filter

The median filter is iteration through the signal window entry by entry, replacing each entry with the median of neighboring entries. Input parameter is **window size.**

Kalmann filter

Implementation of single-dimension Kalmann filter, which is based on iteratively calculating an estimate for measured values in a time series. Following parameters are used to configure the filter: **Measure speed** and **Measure error.** On the first step the Kalmann Gain is calculated as**:**

$$Gain = Estimate\ Error\ /\ (Estimate\ Error + \boldsymbol{Measure\ error}).$$

Initial $Error\ Estimate$ value is 40. Each *New Estimate* as calculated as:

$$New\ estimate = Latest\ estimate + Gain * (RAW\ sensor\ value\ value - Latest\ estimate),$$

$$Estimate\ error = (1 - Gain) * Error\ Estimate + abs(Latest\ Estimate - Curent\ Estimate) * \boldsymbol{Measure\ speed}$$

Finally, the $Latest\ estimate$ value is taken from the $New\ estimate$ variable. Same value is shown to the user as result of the filter output. Then the cycle is repeated from the beginning (new measurement is taken, gain is adjusted, new estimate calculated, new estimate error calculated, etc.).

Selection of a filter type and its configuration type should be based on concrete sensor usage scenario.

Wi-Fi connection settings

These settings allow to configure the Wi-Fi Access Point (AP), which will be created by the readout board. The PC with running SensorMatrixLAB should connect to this AP. When Wi-Fi connection is successfully established, the session can be started over TCP/IP, without the need for a USB connection.

Note that for Wi-Fi AP configuration an existing USB/Ethernet connection is mandatory.

To get current Wi-Fi AP connection configuration, use the **Get Wi-Fi settings** button. The data is then loaded from the board. To update values in **Wi-Fi SSID** and **Wi-Fi Password** fields and upload into the hardware, use the **Set Wi-Fi settings** button.

When Wi-Fi configuration is applied, SensorMatrixLAB software can connect to the hardware over Wi-Fi network created by AP by selecting TCP/IP **Connection type** in **Application Settings.** The **IP address** should be set to 192.168.0.1 and **Port** 1000 should be selected.

In-board frame converter settings

The frame conversion (rearrangement) allows to use the same hardware for reading different printed matrixes with correct data visualization in UI. The in-board frame converter allows frame re-arrangement directly in the Hardware, not on PC side. Thus, the data stream from the hardware can be used afterwards without SensorMatrixLAB, e.g. in embedded applications.

Note that **Hardware-side rearrangement** and **PC-side rearrangement** can interfere with each other. Therefore, please use only one type of rearrangement at the same time.

To get the current in-board frame converter settings use **Read** button. The name of current configuration will be shown in the text field. If **No table** response is received, there is no in-board decoding enabled.

Use the **Write settings** button to apply one of existing frame configuration into the board. When configuration is written into the board, the PC-side frame processing is automatically turned off. To add/delete configurations into available configuration list, refer to section **Settings - Menu “Frame”.**

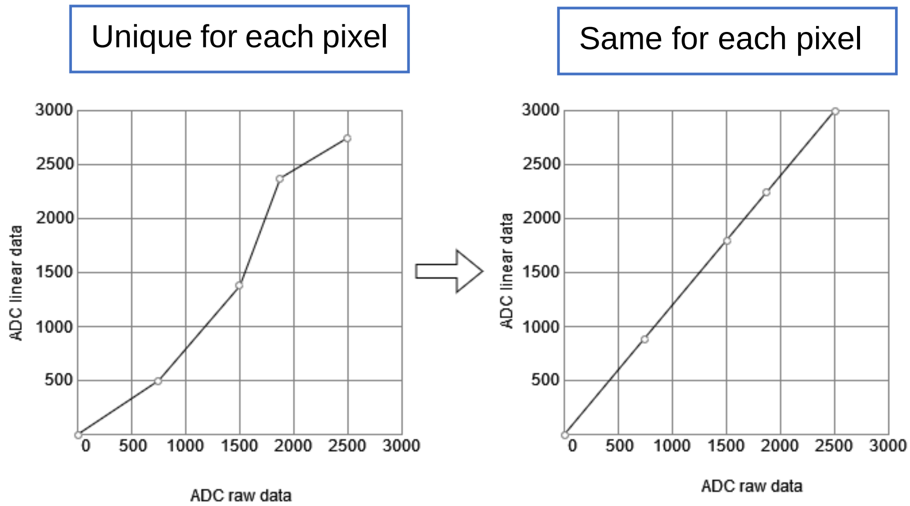
To disable frame processing in the hardware and erase decoding matrix use **Clear Settings** button.

Device Calibration

The calibration feature allows obtaining sensor matrix data in physical units and apply the printed electronics to a variety of applications. It the SensorMatrixLAB and matrix hardware the calibration process is implemented in two consecutive steps:

1. In-board calibration

On the first step the sensor output is being linearized according to pre-loaded calibration curves. Each pixel in the matrix has its own curve as a functional dependency between **ADC raw value** (obtained directly from ADC – Analog to Digital converter) on the board and **ADC linear value.** When the same external stimulus is applied on each pixel in the matrix, they produce different **ADC raw value.** The hardware performs transformation of this initial measurement to **ADC linear value**, which is same for every pixel in the matrix.



Basic idea of in-board calibration process

2. PC-side value mapping

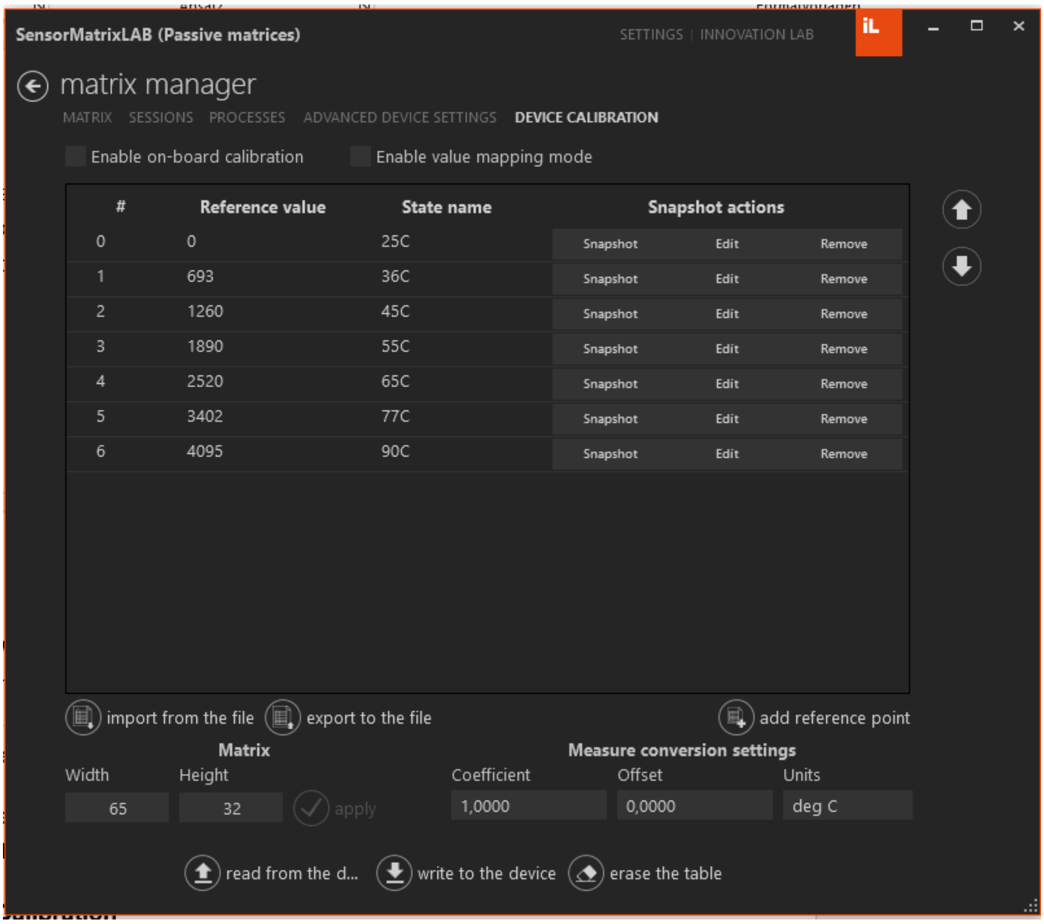
On the second step the values obtained from the hardware are being linearly converted into physical units. This conversion is only performed on the UI side for the visualization purposes. The ADC values are being then processed according to the following formula:

Measurement_in_units = **C * ADC_linear_value + D**,

where **C** and **D** are **Coefficient** and **Offset**, float.

Configuring the In-board calibration

The on-board calibration is a key feature of the sensor calibration process. It can be configured on **DEVICE CALIBRATION** tab of the main window. Note, that the calibration feature is not available on all SensorMatrixLAB licenses.



SensorMatrixLAB – tab "Device Calibration"

In order to configure the **In-board calibration** feature, the following steps should be performed:

1. Make sure, the **In-board frame processing** is configured properly and turned on. PC-side frame processing cannot be used with calibration.
2. (Optional) Read the pre-loaded calibration information from the hardware using **read from the device** button. When necessary, **erase the table**.
3. Select the matrix size. Fields **Width** and **Height** define a size of the reference matrix states for a calibration. Use **Apply** button to save the size.

The matrix size for reference states will be loaded automatically from the last available scan session data. If these values are not correct, check the **In-board frame processing** configuration.

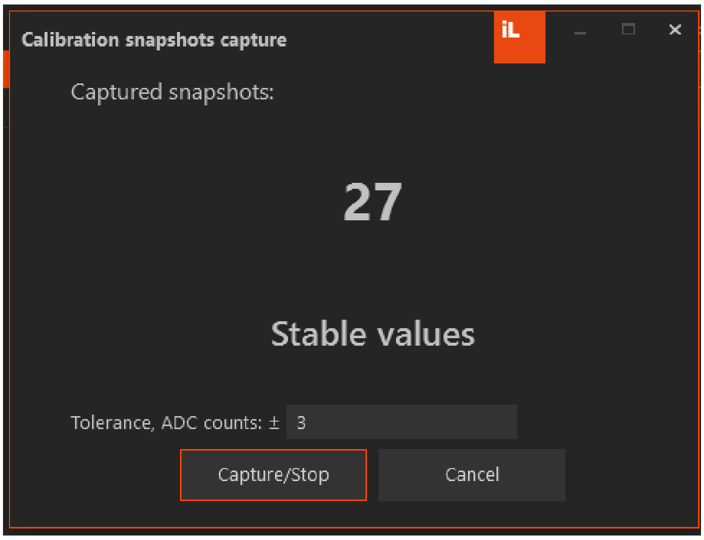
4. (Optional) Load the pre-configured calibration from file on a local drive using **import from the file** button. The file format is *.xlsx containing a calibration information.
5. Create two or more calibration reference points (states). For this an **add reference point** button is used. For a reference state it is assumed, that each pixel within the matrix is in the same condition. This could mean, that the temperature of an entire matrix is same, or the uniform pressure is applied on the whole matrix. When **ADC raw value** of a pixel will be corresponding to one of the reference states, it will be converted to **ADC_linear_value** for this state. For a reference state following important attributes are used:

   Reference value – a target **ADC_linear_value**, which will be calculated for each pixel**,** when its **ADC raw value** is equal to reference state.

   State name – a string attribute allowing user to label the state. Value is not used for computation

   For a reference state following Snapshot actions are available:

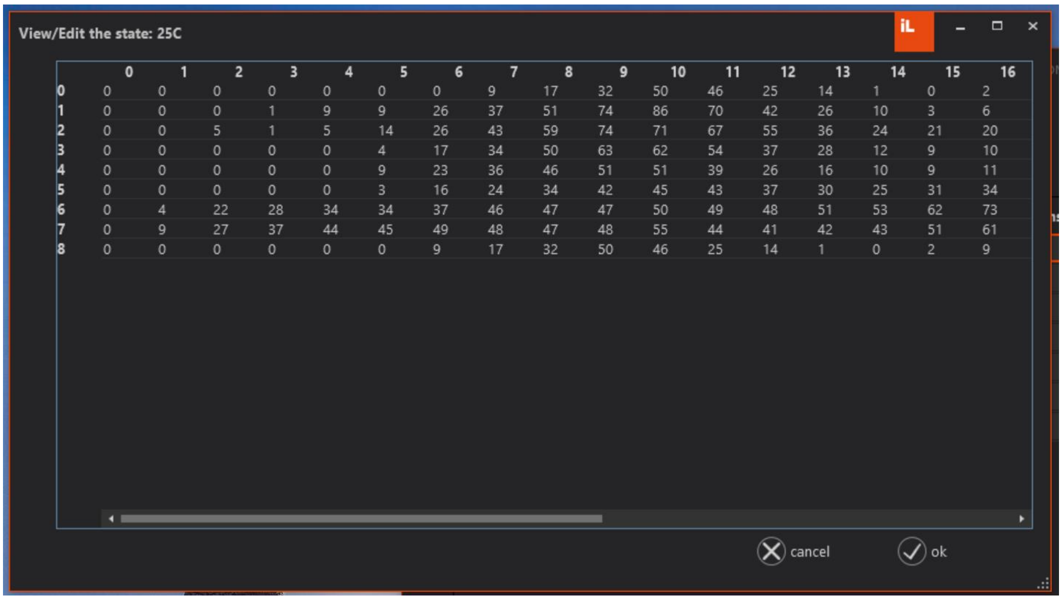
   Snapshot – starts the capture of current matrix state for the reference value. This requires a hardware to be connected to the PC. Use **Start** button to initiate the process. A **tolerance, ADC counts** allows defining a maximum frame-to-frame deviation for every pixel in the scan area (in ADC units), at which the captured data will be considered as stable and usable**.** When enough stable frames are captured, which is indicated by **Stable values** label, the reference snapshot can be saved with **Stop** button. Note, that during data capturing for a reference state the scan settings from **Device Settings** menu are used, including frequency, filtering, delay etc.



Capturing the snapshot data for the reference state

Note, that for a calibration to work minimum one reference state must be created.

Edit – provides a view and edit capability to the captured reference state. In this mode a user can manually adjust the **ADC raw value** for each element in the matrix.

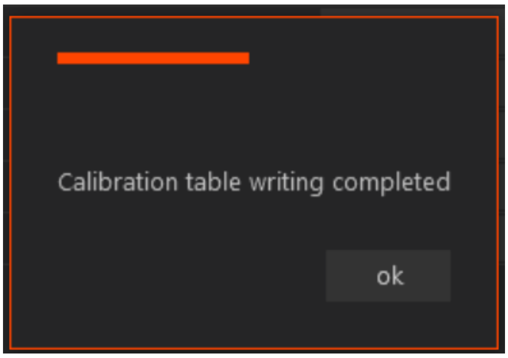


View / Edit the reference state

Edit – deletes the currently selected reference state.

6. (Optional) Export pre-configured calibration into the file for further usage: **export to the file** button. Exported files can be edited manually in MS Excel table processor an imported back into the software.

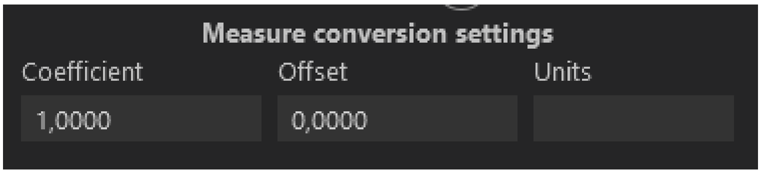
7. Use **write to the device** button to load the on-board calibration settings into the hardware. A message indicates the successful completion.



8. To turn on the feature, use **Enable on-board calibration** flag. If the flag is set, the measured sensor data will be first processed with piecewise linear function, then sent to the PC in the same data format, as for raw (not calibrated) measurements.

Configuring the Value mapping

The value mapping function can be configured on **DEVICE CALIBRATION** tab of the main window. Note, that the feature is not available on all SensorMatrixLAB licenses.



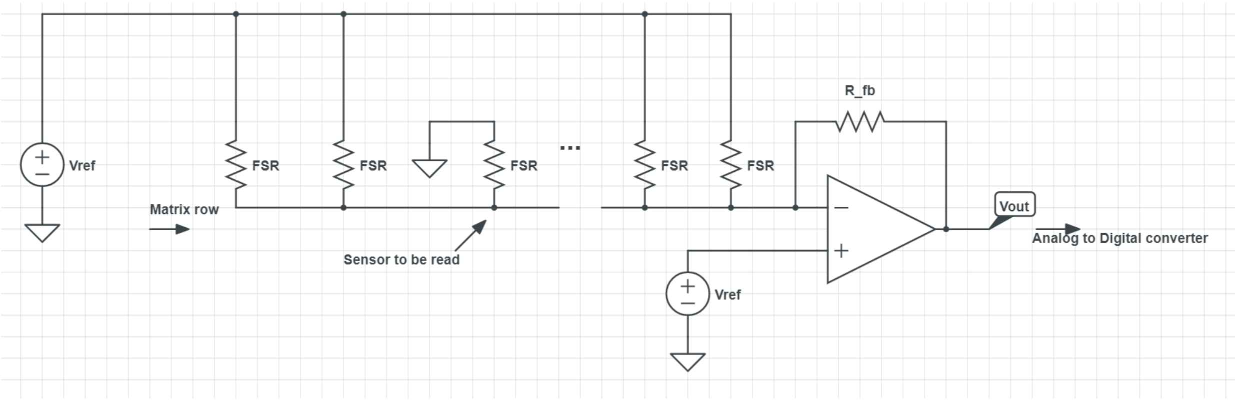
The value mapping can be used together with on-board calibration or without it. To turn on the feature, use **Enable value mapping mode** flag. When on, the data received from the hardware (either calibrated or raw) will be multiplied on the **Coefficient** and summed with **Offset, Units** will be added for the visualization.

Software Revision

Version	Date	Description
3.1.4	15 Sept 2021	Initial public release
3.1.5	22 Feb 2022	• Changes in default UI settings (after installation); • The license is independent of the BIOS version of the PC.
4.0.0	8 March 2022	• Added 32-bit values support into USB / WLAN protocol; • Added non-standard hardware types support; • Sensor calibration added; • Added displaying of current pixel value on mouse hover; • Added displaying up to 16x time-series charts of pixel data; • Frame statistics reworked (added percentiles); • License grades: BASIC / ADV / PRO.

Appendix A. Measurement principle

InnovationLab's crosstalk-less readout electronics use the following schematics for acquiring data from piezo-resistive printed sensor matrices:



Measurement principle of crosstalk-less readout electronics. A single matrix's row is shown.

From this circuit output voltage $V_{\mathbf{out}}$ can be derived as:

$$V_{\mathbf{out}} \approx V_{\text{ref}} + V_{\text{ref}} \cdot \frac{R_{fb}}{\text{FSR}},$$

where $V_{\text{out}} \in [V_{\text{ref}}, V_{\text{Supply}}]$. Before being forwarded to the input of the Analog to Digital converter, a V_{ref} is subtracted from the output V_{out}, resulting in a following formula for sensor signal:

$$V_{\mathbf{adc}} \approx V_{\text{ref}} \cdot \frac{R_{fb}}{\text{FSR}},$$

$V_{\mathbf{adc}} \in [0, V_{\text{Supply}} - V_{\text{ref}}]$, where $V_{\text{ref}} \in [0, 2.5\text{V}]$ and $V_{\text{Supply}} \in [0{,}5\text{V}]$.

As the result of using this circuit, following performance parameters are achieved:

1. No crosstalk between different sensors in a matrix,
2. Sensitivity of the matrix is adjustable: the higher V_{ref}, the higher the sensitivity.

Appendix B. Frame processing with config.json file

In this section it is described how the frame processing fearure of the SensorMatrixLAB works. Following terms are used in the text:
Data frame - a snapshot of data received from the hardware, which corresponds to the state of the sensor matrix at a certain time. Further, in the text, the words "frame" or "data" will mean only ADC measurements (both calibrated or raw) for each pixel on the matrix;

input data, input frame, input data frame - data frame before applying of frame processing;

output data, output frame, output data frame - data frame after applying of frame processing;

frame configuration - a set of parameters that determine how to process the data written to the configuration file;

configuration file - the file accepted for the configuration definition corresponds to the JSON file format (JavaScript Object Notation) with a certain set of fields. It can be downloaded into the measuring board (if this option is supported) or can be used by the client program on the PC side.

Data processing options

The measuring system consists of a hardware and a client program installed on a PC (SensorMatrixLAB). A frame re-arrangement is applied to support different matrix layouts with the same hardware. There are three methods of sensor data frame re-arrangement:

- processing directly in the measuring board (referred in this manual as In-board frame converter);

- processing in the SensorMatrixLAB (referred in this manual as PC-side frame converter);
- no frame processing (both converters are turned off).

Structure of the configuration file

To perform a frame conversion (both in-board or PC-side) the transition matrix is used, which is defining how the incoming data frame will be visualized or processed. This transition matrix is stored in configuration file. The configuration file must contain a certain set of fields for correct processing and must be formatted in accordance with the standard for the JSON file format.

An example for such file *Example.json*:

```
{
    "Configurations":
    [
        {
            "Name":"Example",
            "DriveLines": 4,
            "ScanLines": 8,
            "FrameBits":12,
            "PCM":
            [
            [[94,87],[95,87],[87,87],[93,87],[90,87],[91,87],[88,87],[89,87
            ],],
            [[94,85],[95,85],[87,85],[93,85],[90,85],[91,85],[88,85],[89,85
            ],],
            [[94,83],[95,83],[87,83],[93,83],[90,83],[91,83],[88,83],[89,83
            ],],
            [[94,82],[95,82],[87,82],[93,82],[90,82],[91,82],[88,82],[89,82
            ],]
            ],
            "DisplayRatio":0.5
        }
    ]
}
```

The configuration file can contain several configurations (at least 1), then its format has the following required field **Configurations** with a list of configurations. An example for a configuration file containing two configurations within a file is shown below.

```
{
    Configurations [
    {
        "Name":"...",
        "DriveLines": ...,
        "ScanLines": ...,
        "FrameBits":..,
        "PCM":
        [
        ...
        ],
    "DisplayRatio":...
        }
        ,
        {
        "Name":...,
        "DriveLines": ...,
        "ScanLines": ...,
        "FrameBits":...,
        "PCM":
        [
        ...
        ],
        "DisplayRatio":...
    }
    ]
}
```

Thus, any configuration should contain the following list of fields:

- Name;
- DriveLines;
- ScanLines;
- FrameBits;
- PCM;

- DisplayRatio.

Each of the fields has its own purpose.

Name - string constant, configuration name. In the measurement board, this name is stored only as an identifier of the currently downloaded configuration.

In the SensorMatrixLAB, the configuration name is used as a unique identifier for managing configurations: adding, removing, etc.

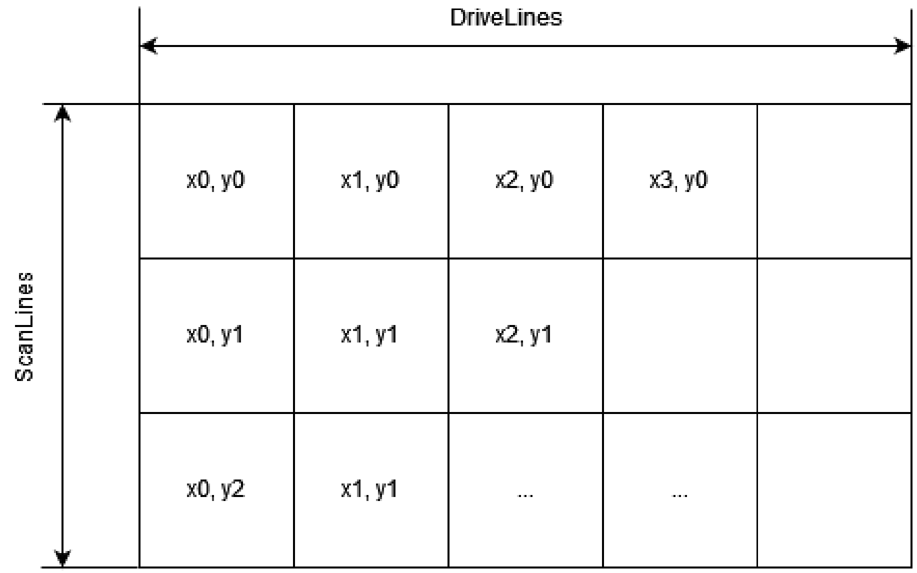
DriveLines: integer constant, corresponds to the number of cells along the width of the frame (X-axis). The minimum possible value is 1.

ScanLines: integer constant, corresponds to the number of cells along the frame height (Y-axis). The minimum possible value is 1.

Thus, fields **DriveLines** and **ScanLines** determine the size of the output data frame after frame processing.

FrameBits: bit resolution of data from the ADC for each pixel on the matrix. The minimum value is 1, the maximum is 32.

PCM: is a two-dimensional array, contains two coordinate values (X and Y) to address cells in input data (a transition matrix). This array can be thought of as a table with a width equal to the DriveLines and a height equal to ScanLines, and each cell contains a pair of values that contain the coordinates of points in the input data frame.



PCM conversion table structure

DisplayRatio: float, the ratio of the width of the frame to the height, is used only in rendering in the GUI if frame processing is enabled. Otherwise, the render ratio is calculated as width-to-height based on the session information received.

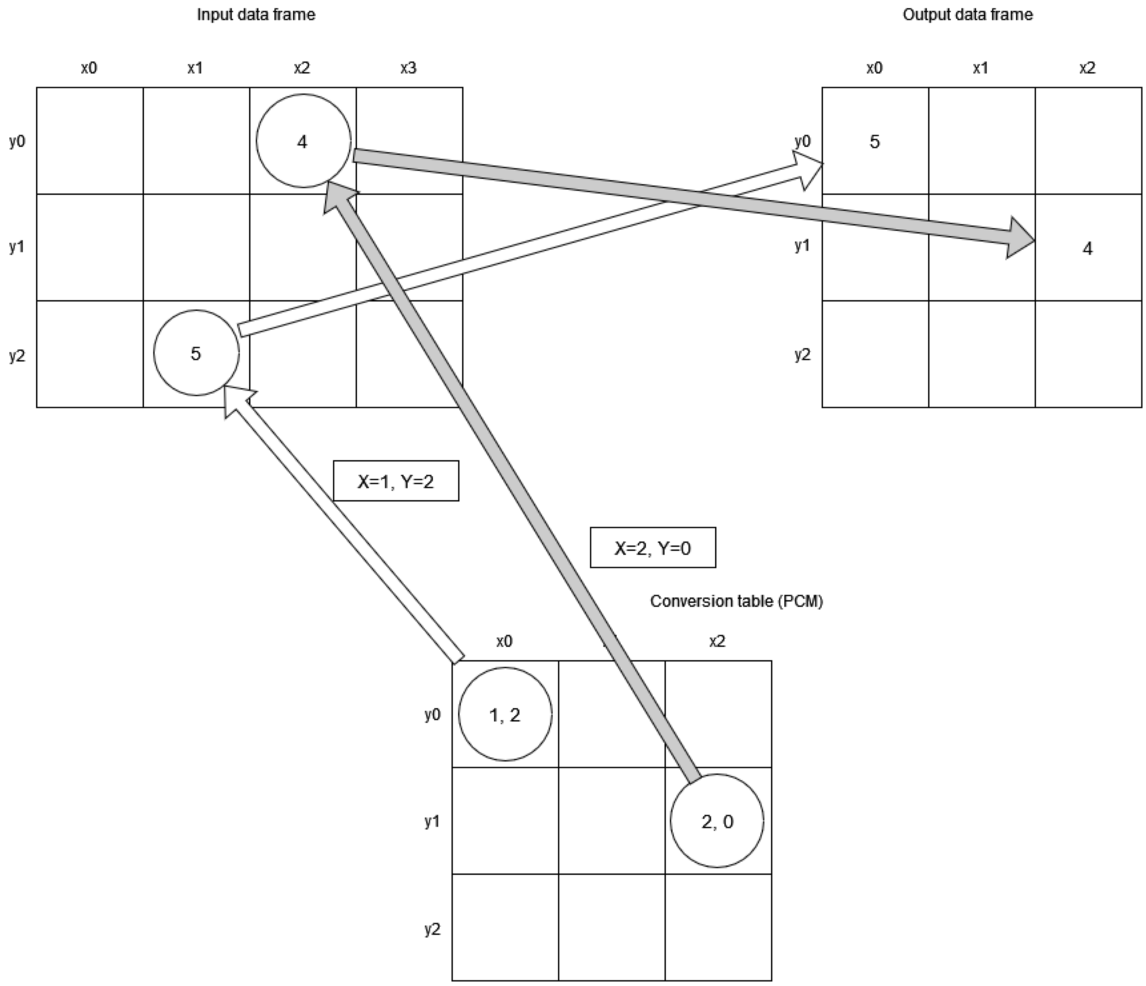
If the configuration is created separately and imported programmatically through the GUI, other fields may appear in the configuration file.

The user can ignore them and should not change them, because these fields are used by the program.

One of these possible fields: **AdcValuesSignType** can be 0 or 1 for a SensorMatrixLAB software version lower than 4.0 and must always be 0 starting at version 4.0.

In this case, a value of 0 indicates that only unsigned integers are stored in the PCM array, and a value of 1 indicates signed integers.

The figure below shows the general principle of frame conversion with a PCM table. The output frame size is always the same as the PCM. Each element of PCM is addressing the pixel data in the input frame. For example, for pixel (x0, y0) in the output data frame a pixel (1,2) is take from the input frame. As the result, the value “5” is takein from input data frame and put into output frame with coordinates (x0, y0). Same process is repeated for the whole PCM table.



General principle of frame conversion with a PCM table

With the frame processing feature user can implement support of any possible matrix layout without introducing changes into the firmware, making it handy to adapt the hardware to a variety of applications.