# 1. TABLE OF CONTENTS

## 2. GENERAL DESCRIPTION

3DHALL SENM3Dx Evaluation Kit version 2.5 features SENM3DxV 2 sensor, replacing earlier evaluation kit versions featuring SENM3DxV1. Starting from August 2024 evaluation PCBs have SENM3DxV2 chip. 3DHall eval kit is designed for evaluation of the real 3-axis magnetic field sensor which is a CMOS integrated device that allows vectoral measurement of the three magnetic flux components Bx, By and Bz (Figure 1) in a field sensitive volume (FSV) of 100 x 100 x 10 µm3. The FSV is about 0.35 mm below the surface of the chip package. *Please read the data sheet of SENM3DxV2 for more details.*

To enable quick sensor testing and operation by users, the kit includes a PCB board with a SENM3DxV2 sensor mounted on it, a control box for sensor communication and configuration, along with a cable and power supply. There is a green LED on the PCB to indicate if the sensor is powered.

Figure 1: Magnetic field vector orientation shows SENM3DxV2 evaluation PCB magnetic field vector orientation:

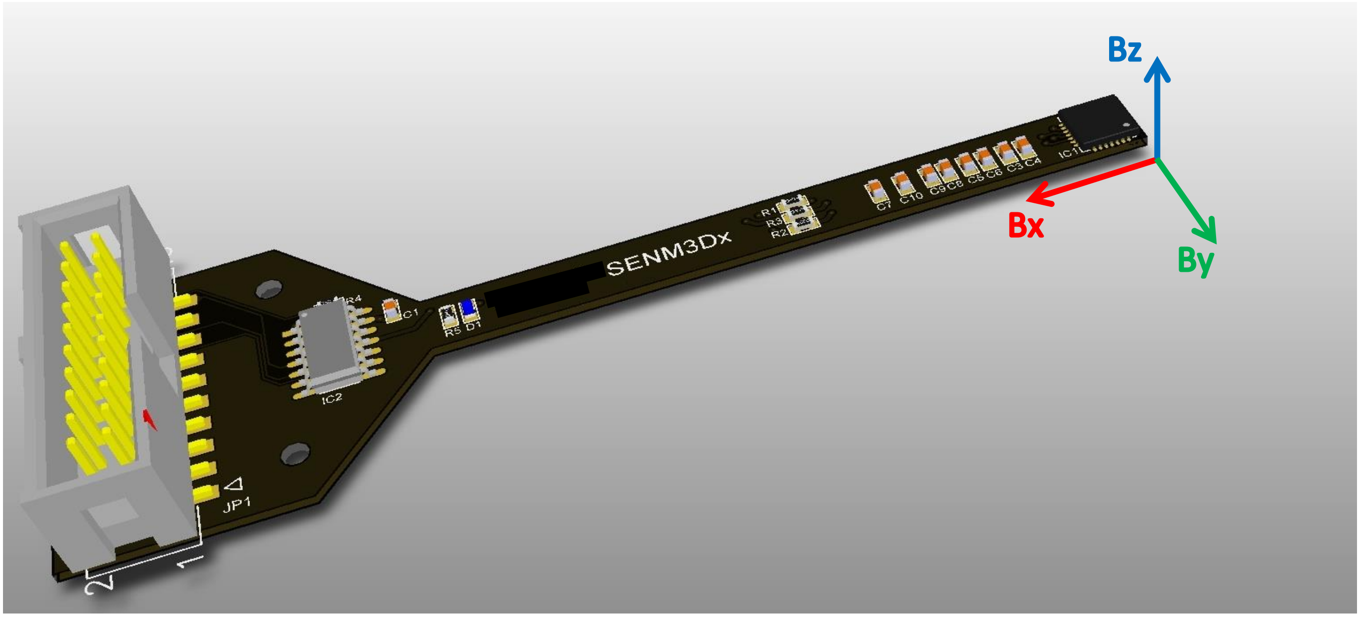


*Figure 1: Magnetic field vector orientation*

Figure 2: Evaluation Kit PCB top-view dimensions shows evaluation PCB dimensions:

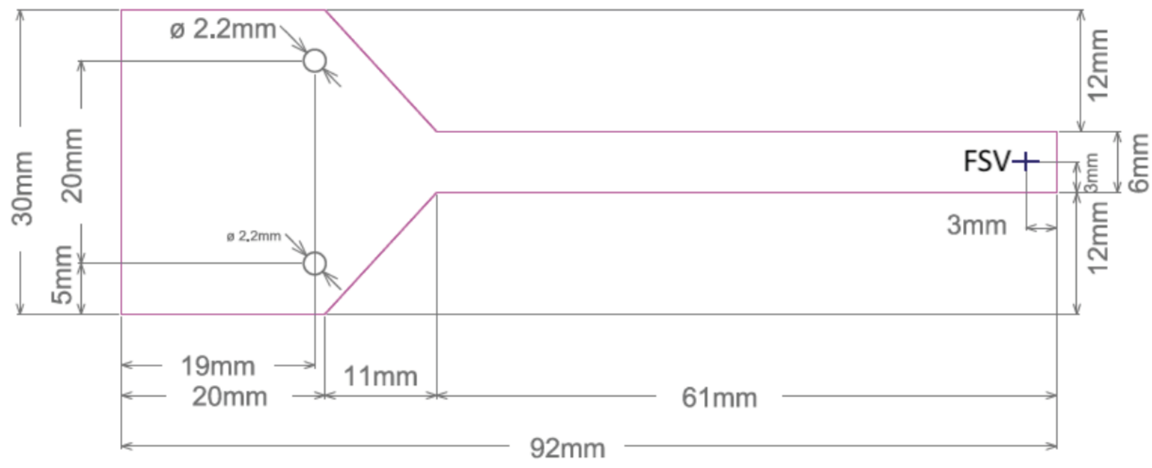


*Figure 2: Evaluation Kit PCB top-view dimensions*

Since SENM3Dx sensors are designed with a variety of functionalities, each iteration may present its own set of compromises and limits as part of our ongoing enhancement process. This current version SENM3DxV2 represents a step forward in our development journey, while we continue to refine and optimize performances.

Figure 3: Vertical position of the field sensitive volume (FSV) Note that these dimensions apply for molded QFN-28. The FSV is about 350 μm beneath the chip surface.

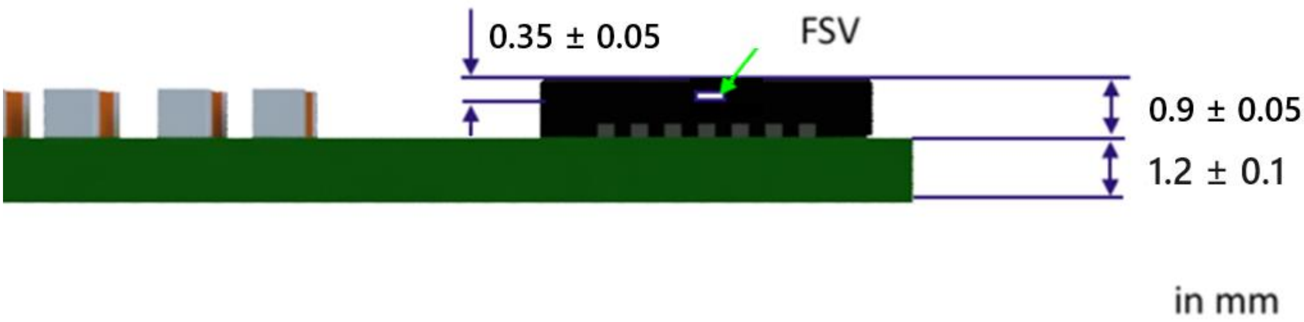


*Figure 3: Vertical position of the field sensitive volume (FSV)*

**Note**: It is recommended not to hot-plug evaluation PCB (i.e. to connect the evaluation PCB if the control box is not powered off) to the evaluation PCB to the control box with the Raspberry Pi (RasPi) embedded computer. So, please make sure that RasPi is switched off (i.e. disconnect the device from power supply).

## 2.1 Evaluation Kit Content

Figure 4: Evaluation kit content: shows items included in the kit. The item number labels correspond to the description in Table 1.

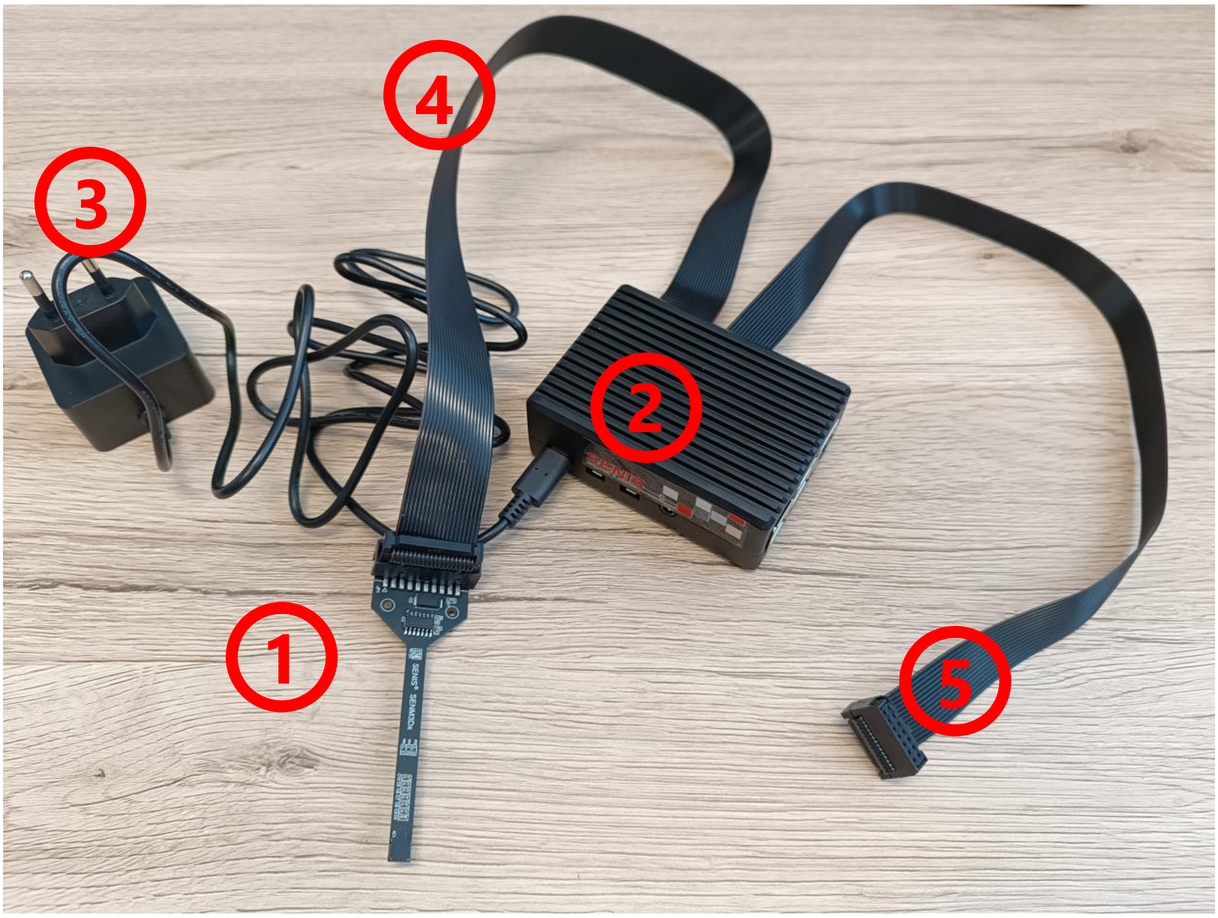


*Figure 4: Evaluation kit content*

| Item No. | Item | Description |
|---|---|---|
| ***1*** | **Evaluation PCB v2.2** | 1.2 mm thick FR4 PCB with SENM3DxV2 sensor mounted |
| ***2*** | **Control Box** | Includes RasPi 4B and provides SPI Interface, API and GUI |
| ***3*** | **Power Supply** | 5.1V original Power Supply with USB C connector |
| ***4*** | **CON01 Connection Cable 01** | 20-way ribbon cable to connect the Control Box (2) to the Evaluation PCB v2.2; IDC connector CON01 |
| ***5*** | **CON02 Connection Cable 02** | 16-way ribbon cable to give access to the analog and digital outputs of the Evaluation PCB v2.2; IDC connector CON02 |

*Table 1: Evaluation Kit Content*

## 3. INTRODUCTION

The small printed circuit board (Evaluation PCB) is equipped with the SENM3DxV2 sensor and its necessary components for stable and reliable operation. All relevant signals are available on a 16 pin (2 x 8) IDC connector (Table 3: IDC connector CON02 pin assignment for evaluation PCB version 2.2 and 2.4). There is no need to install software to operate the sensor since RasPi 4B provides the serial peripheral interface (SPI), advanced programming interface (API) and everything is pre-installed on the microSD card, including the operating system. RasPi is the SPI master and controls the sensor chip. Since SENM3DxV2 is equipped with an EEPROM, the user may use it to write to the sensor memory and operate the sensor without RasPi connected (an external power supply is needed in this case). Note that calibrated SENM3DxV2 on evaluation PCB already have valid EEPROM memory values, therefore it is prudent to read EEPROM memory and save its data before customizing it to ensure that you can revert back to initial state.

The analog outputs (XA, YA, ZA) deliver voltages proportional to the magnetic field and the analog output TA provides a voltage proportional to the chip temperature. These quantities are also available digitally via SPI interface. The three magnetic field components are also additionally offered as pulse-width modulated signals or comparator outputs.

# 4. HARDWARE SET-UP

## 4.1 Connector CON01 – Sensor Adapter PCB and Control Box

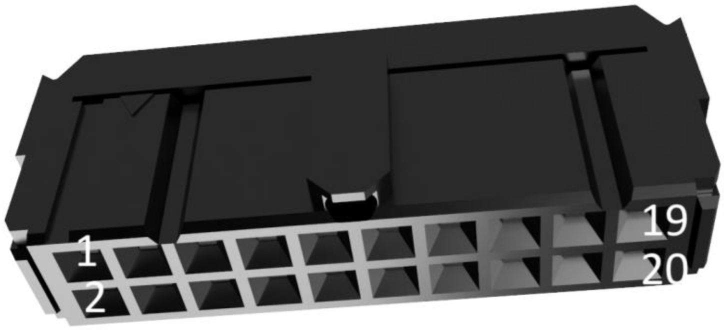


*Figure 5: IDC connector CON01 head female, pitch 2.54 mm, 20 poles*

| CON01 pin | Signal | Description |
|---|---|---|
| *1* | **VREF** | Reference Voltage (+1.25 V); power |
| *2* | **VCM** | Virtual Ground (+2.25 V); output |
| *3* | **GND** | Ground (analog & digital) |
| *4* | **AMUX** | Analog Output ASIC Temperature |
| *5* | **ZA** | Analog Output Bz |
| *6* | **YA** | Analog Output By |
| *7* | **XA** | Analog Output Bx |
| *8* | **ZD** | Digital Output Bz (PWM or comparator) |
| *9* | **VCCA** | Internal regulated analog supply voltage (4.5V); power |
| *10* | **TEST_EN** | For internal use only; do not connect |
| *11* | **YD** | Digital Output By (PWM or comparator) |
| *12* | **XD** | Digital Output Bx (PWM or comparator) |
| *13* | **DMUX** | For internal use only; do not connect |
| *14* | **VDD** | Internal Core Supply Voltage (+3.3 V); power |
| *15* | **VCC** | Main Supply Voltage of the adapter PCB incl. SENM3Dx sensor(+5V); power |
| *16* | **GND** | GND; power |
| *17* | **MOSI** | SPI Interface Master Out Slave In Signal |
| *18* | **MISO** | SPI Interface Master In Slave Out Signal |
| *19* | **MCLK** | SPI Interface Clock Signal |
| *20* | **SSB** | Chip (Slave) Select (active-low) |

*Table 2: IDC connector CON01 pin assignment for evaluation PCB version 2.2, 2.4 and 2.5*

## 4.2 Connector CON02 – Analog and Digital Signals

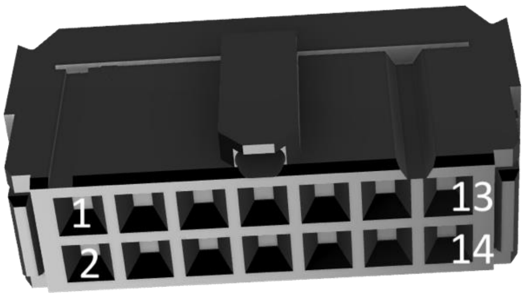


*Figure 6: IDC connector CON02 head female, pitch 2.54 mm, 14 poles*

| CON02 pin | Signal | Description |
|---|---|---|
| *1* | **VREF** | Reference Voltage (+1.25 V); power |
| *2* | **VCM** | Virtual Ground (+2.25 V); output |
| *3* | **GND** | Ground (analog & digital); power |
| *4* | **AMUX** | Analog Output ASIC Temperature TA |
| *5* | **ZA** | Analog Output Bz field component |
| *6* | **YA** | Analog Output Bx field component |
| *7* | **XA** | Analog Output By field component |
| *8* | **ZD** | Digital Output Bz (PWM or comparator) |
| *9* | **VCCA** | Internal regulated analog supply voltage (4.5V); power |
| *10* | **TEST_EN** | Digital Input; For internal use only; do not connect |
| *11* | **YD** | Digital Output By (PWM or comparator) |
| *12* | **XD** | Digital Output Bx (PWM or comparator) |
| *13* | **TEST_DMUX** | Digital Output; For internal use only; do not connect |
| *14* | **VDD** | Internal Core Supply Voltage (+3.3 V); power |

*Table 3: IDC connector CON02 pin assignment for evaluation PCB version 2.2, 2.4 and 2.5*

## 4.3 Schematics of the evaluation board

Figure 7: Schematic of the evaluation PCB version 2.2 shows the schematic of the evaluation PCB. Evaluation PCB version 2.4 and 2.5 have the identical schematic.

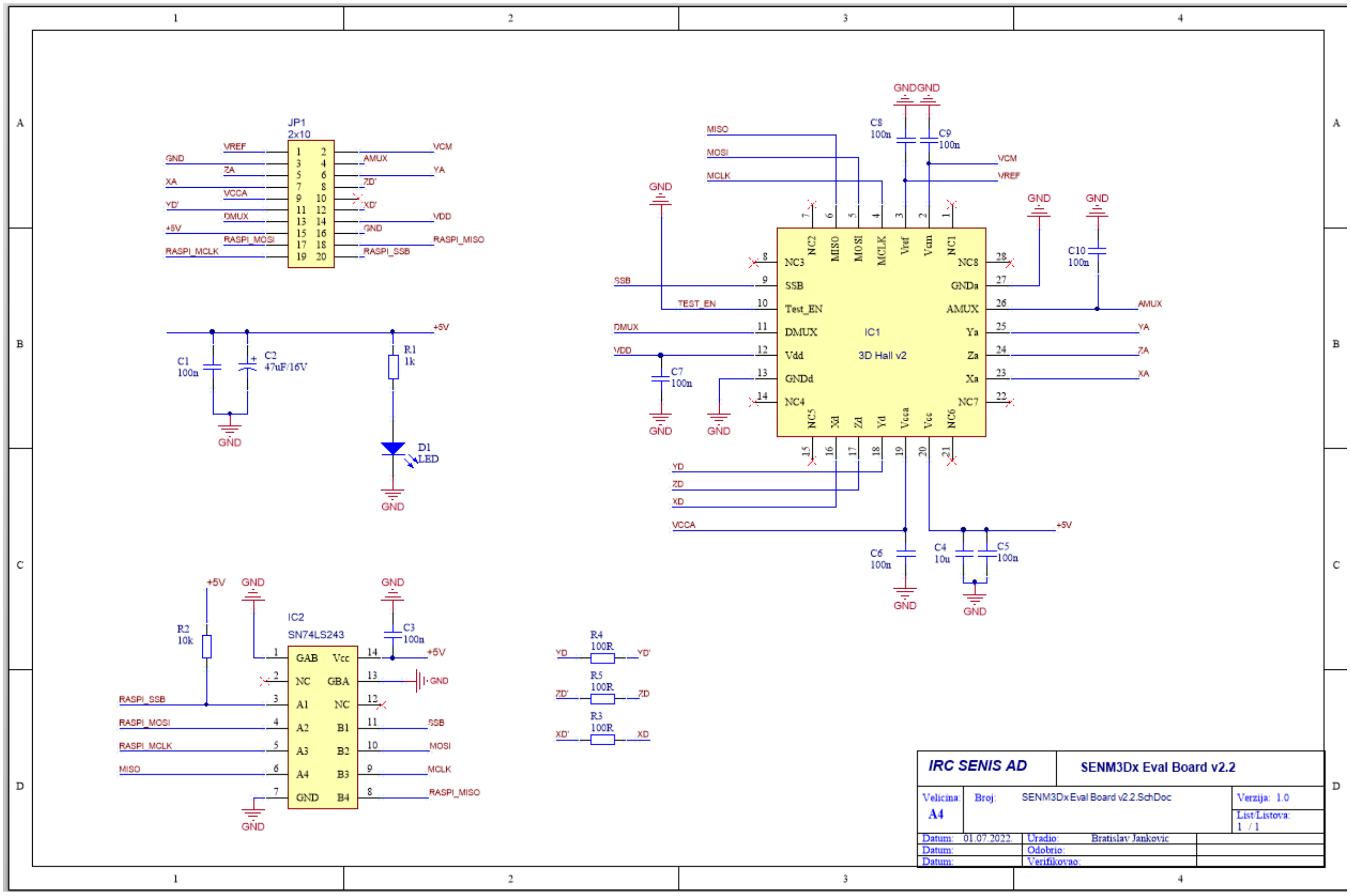


*Figure 7: Schematic of the evaluation PCB version 2.2.*

## 5. SOFTWARE

The RasPi is equipped with a 32 GB microSD card with all necessary software pre-installed and tested for Raspbian GNU/Linux version 10 (buster)[1] and python version 3.7.3[2]. Note that updates or upgrades of the system may cause compatibility issues and affect the functionality and/or performance of the software and libraries provided by SENISENS.
Figure 8: Shows the organization of the files which are essential for the communication with SENM3Dx located at /home/pi/software/threeDhall_sw.

```
.
├── config
│   ├── calibration.cfg
│   ├── calibration_generic.cfg
│   └── main.cfg
├── init_interface.py
├── __init__.py
├── pics
│   ├── hall_icon.png
│   ├── pcb_icon.png
│   └── senis_logo.png
├── __pycache__
│   └── ThreeDHALLInterface.cpython-37.pyc
├── README.md
├── ThreeDHallGUIv7.py
├── ThreeDHALLInterface.py
└── utilities
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-37.pyc
    │   └── utilities.cpython-37.pyc
    └── utilities.py
```

*Figure 8: Illustrative file structure of SENISENS software located at /home/pi/software/threeDhall_sw*

[1] https://www.raspberrypi.org/documentation/raspbian/
[2] https://www.python.org/downloads/release/python-373/

## 5.1 Graphical User Interface

The graphical user interface (GUI) is part of the pre-installed SENISENS software and provides basic functionality to read data from the SENM3Dx sensor and is organized in three tabs: Settings, Plot, Save.

The **Settings tab** (Figure 9) allows to control the measurement range (gain) selection for all channels (Bx, By, Bz), switching channels on and off, adjusting the offset of the Hall element directly (trough adjusting the bias current) and verify the SPI interface. At any time, the current chip temperature and magnetic field reading is acquired from the sensor and displayed.

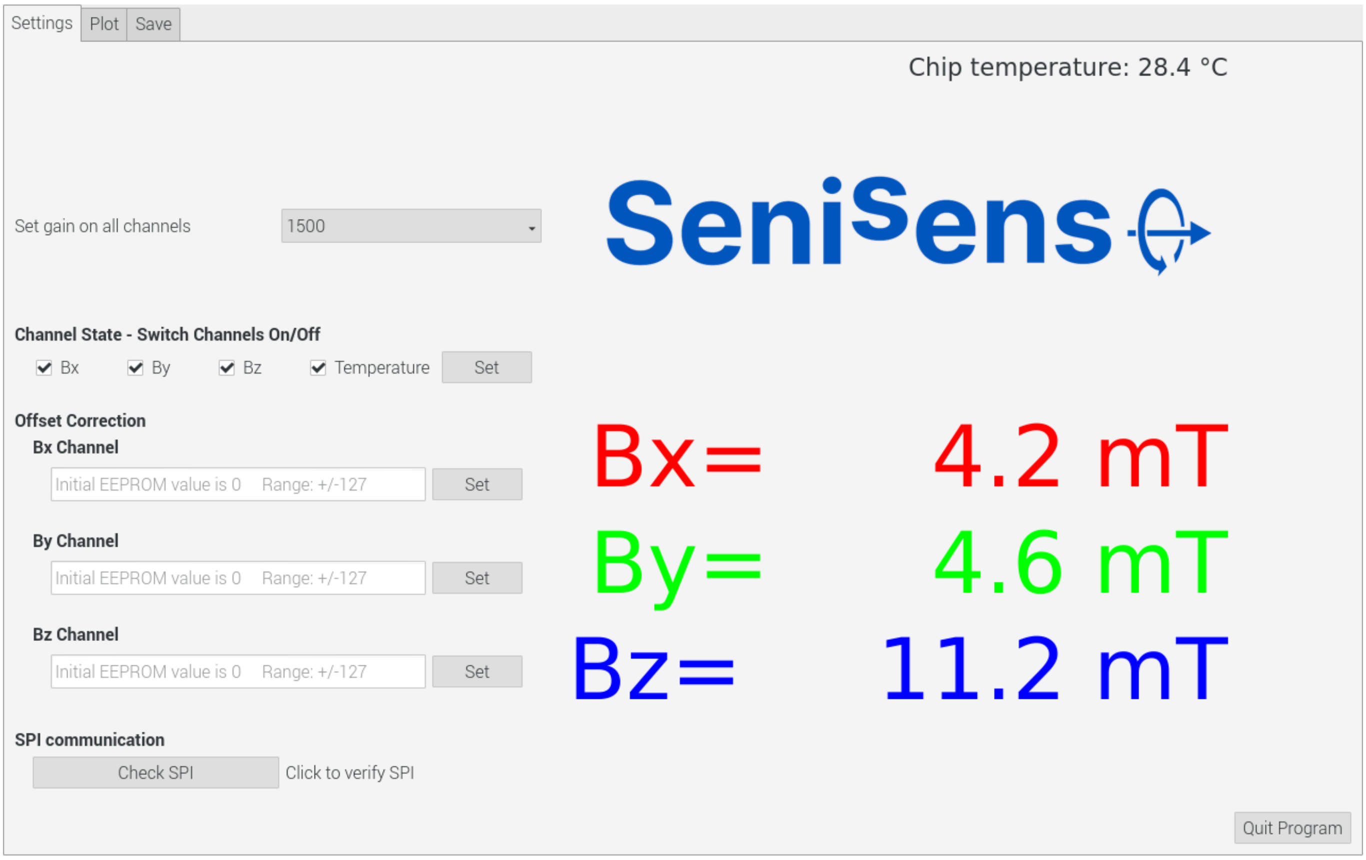

*Figure 9: Image of the settings tab*

The **Plot tab** (Figure 10) allows the user to see the magnetic field reading over time. If the window is enlarged, it is a very convenient way of looking live at measurement signals.

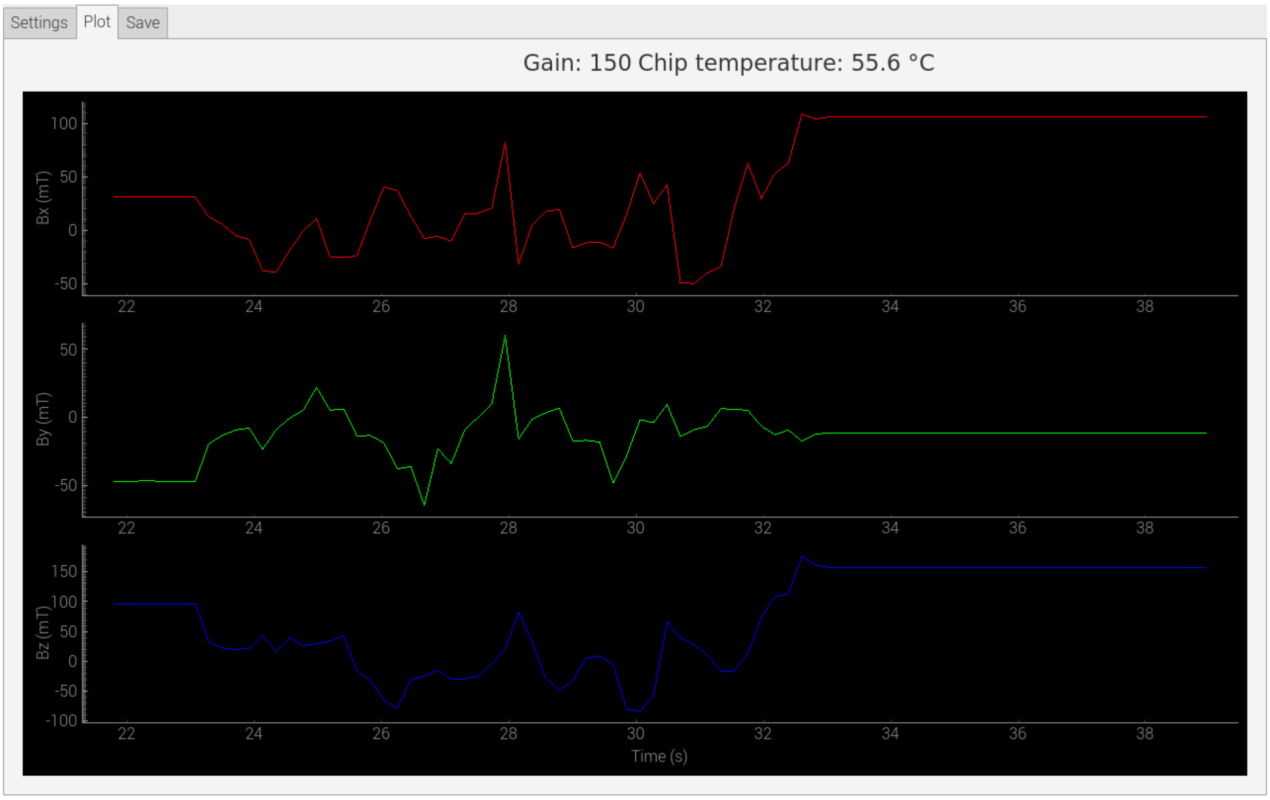


*Figure 10: Image of the plot tab*

The **Save tab** (Figure 11) consists of:

- A text field to define the file path and name, which is also possible by clicking on the symbol next to it.
- Save and Stop buttons which allow to start and stop data saving on-the-fly.
- The save n number of samples button with a text field, which allows to save a certain number of data points.
- A plot area to show the sampled data, which is saved to a .csv file.

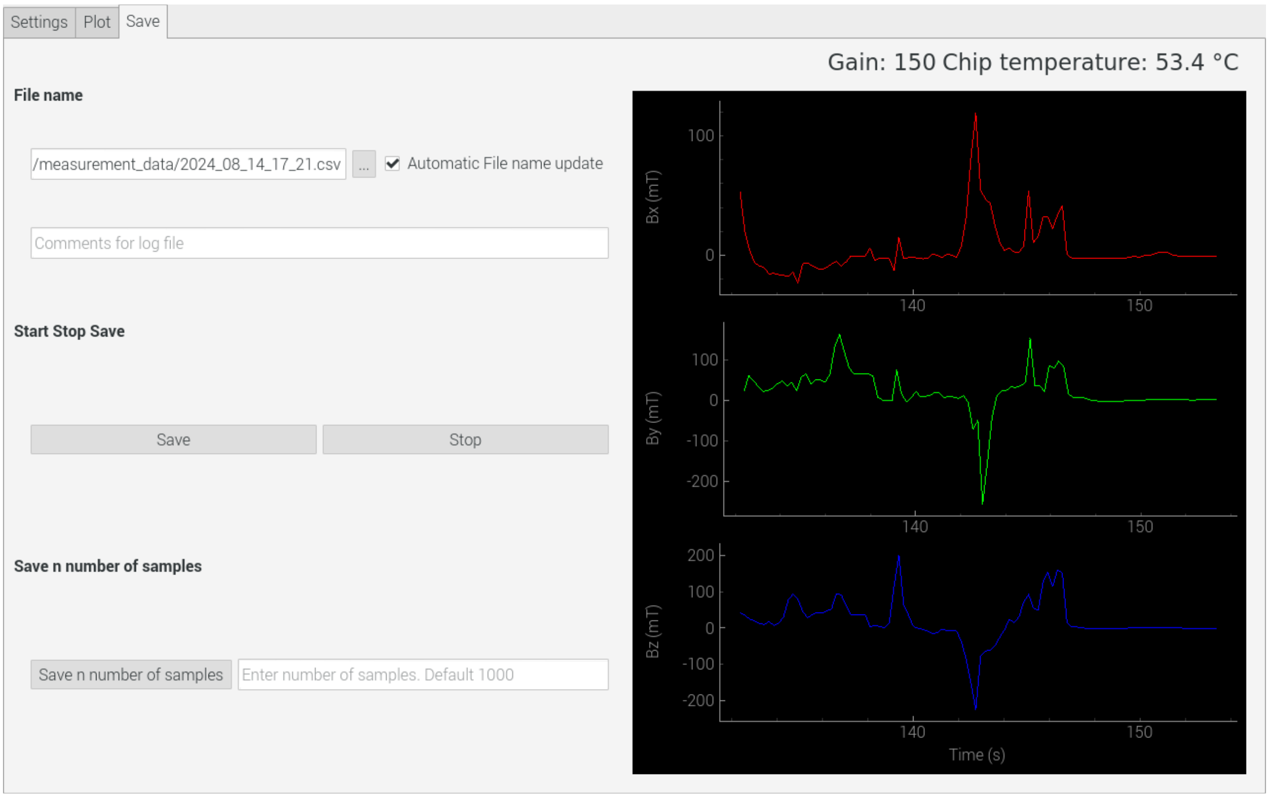


*Figure 11: Image of the save tab*

Figure 12 shows an example of a saved data file in .csv format.

2020_03_16_10_26.csv

```
Comment:
Gain: 150
Offsets
Bx axis: -100
By axis: -10
Bz axis: 20
Gain: 150
Offset correction values: Bx = -100, By= -10, Bz= 20
Time (s)  ,Bx (mT)  ,By (mT)  ,Bz (mT)  ,Temp (degC)
120.621   ,2.1      ,24.8     ,17.5     ,46.2
120.835   ,2.1      ,24.7     ,17.5     ,46.2
121.049   ,2.1      ,24.7     ,17.6     ,46.1
121.268   ,2.2      ,24.8     ,17.5     ,46.1
121.481   ,2.2      ,24.7     ,17.5     ,46.1
121.694   ,2.2      ,24.8     ,17.5     ,46.1
121.907   ,2.1      ,24.7     ,17.6     ,46.1
122.120   ,2.2      ,24.8     ,17.6     ,46.1
122.333   ,2.2      ,24.8     ,17.6     ,46.1
122.546   ,2.2      ,24.7     ,17.6     ,46.1
122.758   ,2.2      ,24.8     ,17.6     ,46.1
122.971   ,2.1      ,24.8     ,17.6     ,46.1
123.185   ,2.2      ,24.7     ,17.6     ,46.1
123.399   ,2.2      ,24.8     ,17.6     ,46.1
123.612   ,2.2      ,24.8     ,17.6     ,46.1
123.836   ,2.2      ,24.8     ,17.6     ,46.1
124.050   ,2.2      ,24.8     ,17.6     ,46.1
124.263   ,2.1      ,24.8     ,17.6     ,46.1
124.477   ,2.2      ,24.8     ,17.6     ,46.1
124.691   ,2.1      ,24.8     ,17.6     ,46.1
```

*Figure 12: Image of a saved .csv file*

### 5.1.1 Measurement and Log File Location

Measurement files are saved per default to the file location /home/pi/data/measurement_data/ and log files are saved to /home/pi/data/logs/ in time stamped folders.

## 5.2 API

Since the GUI supports a limited set of the SENM3Dx programming possibilities, the user gets access to the full sensor's functionality through the advanced programming interface (API). The API is implemented in Python as interface class in the file ThreeDHALLInterface.py.
Either use the link "Start 3DHALL Interface" on the desktop to run iPython3 and invoke the interface, or do it manually, i.e. open a shell and type: cd /home/pi/software/threeDhall_sw && ipython3 -i init_interface.py
This will create an object "threeDhall_dev" of the interface class, and the user is ready to access all methods and class variables. The most important methods are listed alphabetically and are explained in the following section.
Looking at the init_interface.py file should give an idea how the user can just create his own Python file and insert code to perform certain tasks.

### 5.2.1 Most Important API Methods

Initialise the interface class to get access to the API:
threeDhall_dev = threeDhall.ThreeDHALLInterface()
The following methods are then accessible with the "dot" operator e.g.: threeDhall_dev.closeSPI()
Note that the utility methods, such as write_bits_to_byte, are not explained here. Often functions are shown with their default values for arguments e.g. configureSPI(SPIMaxSpeedHz="7.8MHz").

**activate_EEPROM_config()**

The method writes the key and the actual checksum of the memory content to the EEPROM. After power cycle the sensor chip, the EEPROM data should be present in the respective registers.

**change_channel_state(state=["on","on","on","on"], channel=["x","y","z","t"])**

Set the Bx, By, Bz and Temperature channel to the respective state. All four channels have to be in the parameter list. Valid states are "on" and "off". Use print_status() to see the effect of the changes.

**closeSPI()**

Close the connection to the SPI interface of the RasPi.

**configureSPI(SPIMaxSpeedHz="7.8MHz")**

Set the SPI interface clock of the RasPi and mode (CPOL, CPH), data width, etc. according to the SENM3Dx interface definition. Valid clock speed parameters are: '7.6kHz', '15.2kHz', '30.5kHz', '61kHz', '122kHz', '244kHz', '488kHz', '976kHz', '1.9MHz', '3.9MHz', '7.8MHz', '15.6MHz', '31.2MHz', '62.5MHz', '125MHz'
Note that the preset SPI clock setting is '7.8MHz' for stable operation provided ribbon cable. If longer cable is used, SPI clock might need to be reduced to ensure proper communication. Similarly to this, SPI clock can be increased if shorter connection is made between SENM3Dx and SPI controller.

**deactivate_EEPROM_config()**

Write 0x00 as key and a wrong checksum, so that data will not be loaded from EEPROM at power-up.

**erase_EEPROM()**

Write 0x00 to the entire EEPROM.

**get_gain()**

Returns the actual gain setting as integer value which corresponds to the measurement range.

**get_temperature()**

Returns the actual chip temperature in degC as float number.

**openSPI(SPIBusID=0, SPIDeviceID=0)**

Opens the SPI interface with default values SPIBusID=0 and SPIDeviceID=0, which corresponds to the wiring of the evaluation PCB. The RasPI would be able to handle two SPI devices on bus 0 (SPIDeviceID=0 & 1).

**print_status()**

Print out the actual state containing: Hall element bias currents, channel status, chip temperature, EEPROM key and checksum status, etc.

**readSPI_clk()**

Returns the current SPI clock setting as integer number.

**read_EEPROM_check_sum_reg()**

Returns the current value stored at address 0x1FF, where in case of EEPROM data loading active, the valid check sum is expected.

**read_EEPROM_data(start_addr=0x00, num_bytes=1)**

Read and return the number of bytes as list, beginning from start address in the EEPROM.

**read_EEPROM_key_byte_reg()**

Returns the current value stored at address 0x1FE, where in case of EEPROM data loading active, the valid key value 0xA5 is expected.

**read_EEPROM_reg_val(reg_address=0x0A)**

Read and return EEPROM data from a single address in the memory. Note that the EEPROM address offset is not used (e.g. EEPROM address 0x10A is equal to reg_address=0x0A).

**read_adc_reg_data()**

Returns a list of the four raw 16 bit ADC values of [Bx, Bz, By, Temperature].

**read_b_field_values(num_of_samples=1, read_temperature=True)**

Read and return a list of num_of_samples ADC values converted to magnetic field units mT. Note that the temperature value is always read and may be removed from the return value by setting the read_temperature parameter to False.

**read_print_EEPROM_content()**

The function reads, prints and returns the entire EEPROM content values from address 0x100 to 0x1FF as list.

**read_reg(reg_address=0x09, num_of_bytes=1)**

Basic read register read function which returns a list of num_of_bytes values from reg_address on. Note that also the three first zero bytes which are replied by the sensor are returned. So, if we perform for instance read_reg(reg_address=0x09, num_of_bytes=3), the function returns: [0, 0, 0, u, v, w], where u is the value at address 0x09, v of 0x0A and w of 0x0B.

**read_reg_val(read_reg_val(reg_address=0x09)**

Returns only the register value of reg_address.

**select_digital_output(sel=["comp","comp","comp"], channel=["x","y","z"])**

The output selection can be either the threshold/comparator output "comp" or "pwm" for each channel individually. Note that sel and channel has to be a list with three items each and only the key word "pwm" will cause the output to change from the comparator one. Use print_status() to see the effect of changes.

**select_spinning_phases(phases=4)**

Select either phases=2 or 4 to set the number of spinning phases. Use print_status() to see the effect of the changes.

**set_amplifier_gain(data=150)**

This function sets all three channels to the same gain. Accepted values for data=3000, 1500, 150 and 15 as integer which reflects the measurement ranges of 30 mT, 60 mT, 500 mT and 2T for SENM3DxV1 and 20mT, 40mT, 400mT and 4T for SENM3DxV2.

**verify_EEPROM_data_for_config()**

Returns "True" if the EEPROM content checksum is OK and the key is valid. "False" is returned otherwise.

**verify_spi_interface(iterations=1000)**

Data is written and read for multiple iterations and if it matches, "True" is returned. Otherwise the SPI interface has to be considered as corrupt, and "False" is returned.

**write_EEPROM_reg(reg_address=0x10, data_to_write=0x00)**

Write byte value as parameter data_to_write to EEPROM address at reg_address.
write_default_values_to_EEPROM

**write_reg(reg_address=0x10, data_to_write=0x00)**

Write byte value (data_to_write) to any register address (reg_address) from 0x00 to 0xFF. Note that all register write functions (in contrast to EEPROM access) are based on write_reg() and returns "True" if the write access was successful, "False" otherwise.