Instructions to setting up the

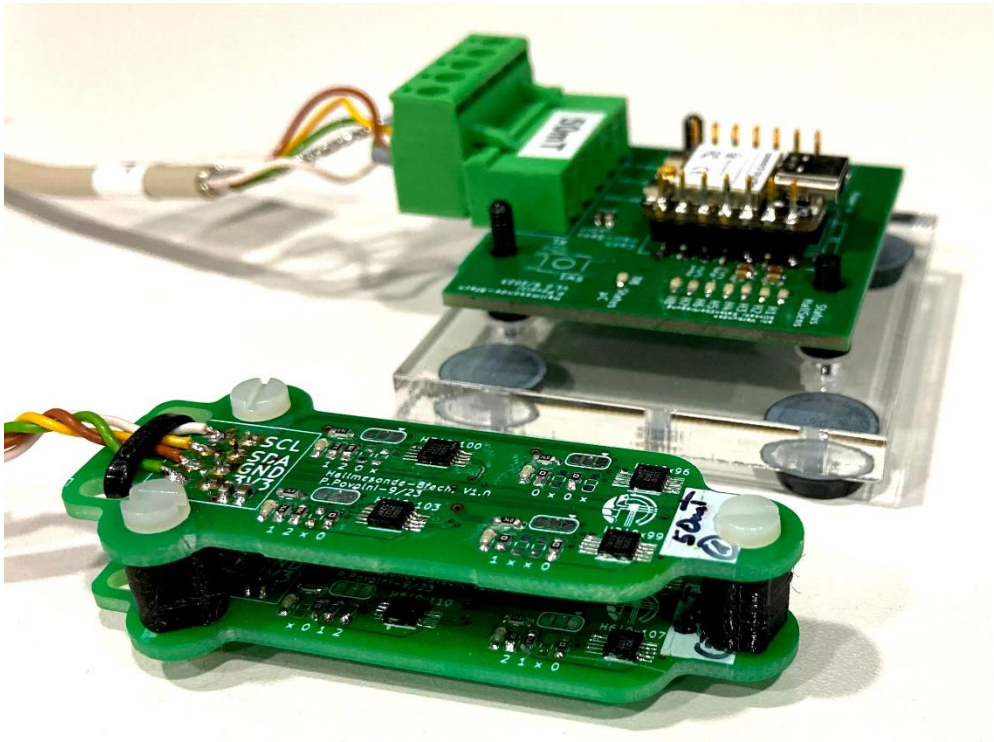
Hall sensor

Project

Easy Scalable, Low-Cost Open Source Magnetic Field Detection System for Evaluating Low-Field MRI Magnets using a Motion Tracked Robot

Pavel Povolni

October 2024



Content

1 Preamble

These instructions describe the installation, commissioning and calibration of the Hall sensor.
Further information can be found in the publication, as well as in the Appendix and Supplementary Information referenced there.
Software used:

- PCB design: KiCad 7 (Free)
- Mechanical design (CAD): Autodesk Inventor 2022 (Commercial). All parts exported as .Steps
- Embedded software: Arduino IDE 2.2.1 (free)
- Calibration: Matlab 2021b (w/o expensive toolboxes), FEMM 4.2 (free)

If you have any further questions, please contact us!

2 Hardware

2.1 Assembly PCB's

The boards were developed in the open source software KiCad 7.0.
The design files can be found under: ...\1_HallSensor\11_Hardware\111_PCB File.
Opening the file "Hall_Probe_8Ch.kicad_pro" opens the project and allows you to edit the schematics ("Hall_Probe_8Ch.kicad_sch") and the layout ("Hall_Probe_8Ch.kicad_pcb").

The entire Hall sensor consists of 3 individual PCBs, which were connected in the layout to form a single PCB (cheaper production of the PCB itself). The boards must therefore be separated before assembling the parts.

The board is designed in 2 layers (F/B) with the following design rules:

- Minimum clearance 0.1mm
- Minimum track width 0.2mm
- Minimum via diameter 0.4mm
- Minimum residual ring 50µm
- Minimum hole 0.2mm

Our PCB was produced by the manufacturer MultiCB (www.multi-circuit-boards.eu/en). The PCB (.kicad_pcb) can be uploaded directly there. Depending on the desired quantity, the manufactured PCB costs beginning at 15€. We have also had very good experiences with the manufacturer JLCPCB (www.jlcpcb.com). There, the layout must first be converted to a Gerber file and then uploaded (https://jlcpcb.com/help/article/how-to-generate-gerber-and-drill-files-in-kicad-7)
Production is much cheaper there (6€ for 5 pieces), but you have to pay by credit card.

All necessary electronic components are listed in an Excel list:
...\1_HallSensor\11_Hardware\PartList.xlsx

We have ordered all components from the supplier Mouser Electronics (www.mouser.com). The supplier number is also shown in the list. We have made sure that none of the components have any special (& expensive) tolerance requirements. Only the Hall sensor itself and the microcontroller must be of the specified types, so that the software will work.

2.1.1 General

- *Preparation*

The PCB must be sawn into individual boards before assembly. Figure 1 shows the cutting lines. A chain of via's makes it easier to break the boards off from each other. In our case, however, a fine metal saw was better and created a more controlled break. For a nice result and blunt edges, it is worth removing the remnants with a metal file.
You could also assemble the PCBs before sawing. However, great care must then be taken to ensure that no mechanical loads/vibrations act on the solder joints during sawing (otherwise they will break).

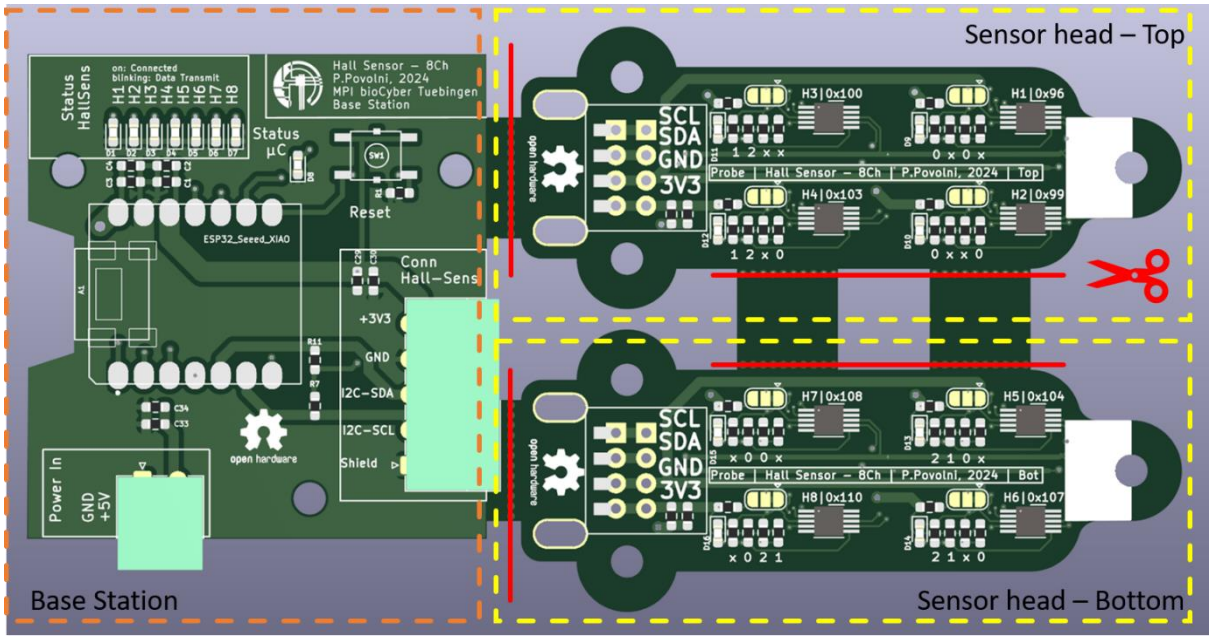


Figure 1: CAD rendering of the board (Microcontroller ESP32 is not shown)

- *Soldering the Sensor Head*

Both parts of the Sensor Head (top/bottom) are assembled almost identically. The only difference is the addressing of the Hall sensor ICs, which is possible via 4 resistors each (see chapter 2.1.2).
These PCBs have parts on both sides so that the head is as small as possible.

First you should solder the Hall sensor ICs. Take your time, because the footprint (DFN10) is challenging:
The pins are mainly located under the housing and a pad in the middle is used for mechanical attachment (no electrical function). For this IC, soldering with hot air and a lot of flux is recommended. The pins can also be contacted on the side. With plenty of flux and a chisel-shaped soldering tip, they can be contacted with the large pads on the PCB. A short circuit can be recognized optically with a microscope/magnifying glass. There are good tutorials on YouTube how to solder such parts.

SMD footprint 0603 was used for capacitors/resistors/LEDs. With a little practice and a fine soldering tip (possibly a magnifying glass/microscope for checking), these can be soldered well. Take your time if you're not trained & don't worry it's easier to solder such small parts than expected. The SMD jumpers (gold-colored footprints in Figure 1) were added to the schematic to change the "interrupt" mode of the Hall sensors if necessary. However, the interrupt was never used, so the jumpers can be removed in a future version.

- *Soldering the Base Station*

For the Base Station, the microcontroller (XIAO SEEED based on ESP32C3) should be soldered first. There are 2 options (see Figure 2): Either the microcontroller is soldered on directly (b, option 2) or SMD pin headers (e.g. Mouser: 855-M20-8770746 , Harwin M20-8770746) (b, option 1) are used. The pinheaders are not really necessary. But maybe it makes sense for you if the microcontroller can be replaced in a plug&play-way (then you would need to add female pin headers to the Microcontroller).

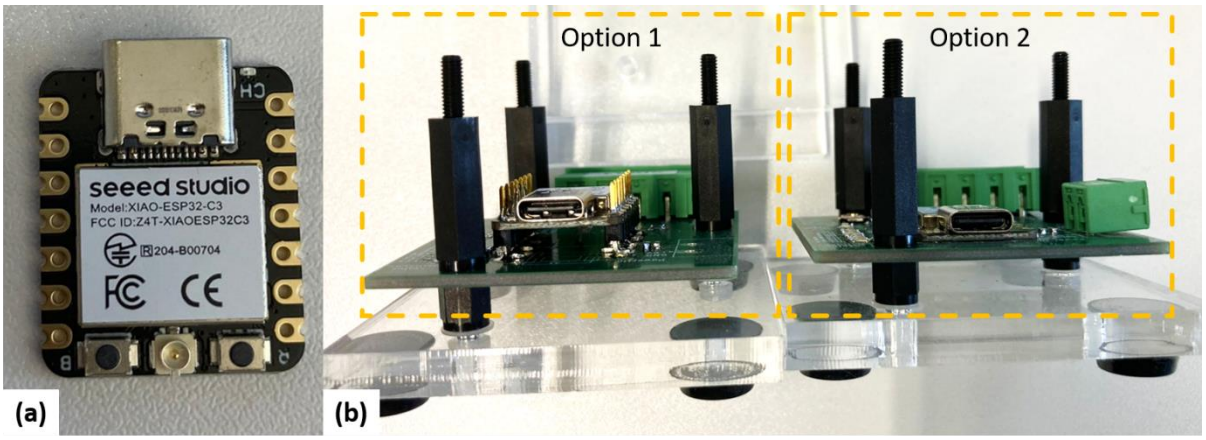


Figure 2: (a) Microcontroller used (seeed XIAO-ESP32-C3). (b) Showing the 2 options how the microcontroller can be soldered.

The button SW1 is used to reset the microcontroller. The microcontroller itself also has a reset button, so the extra button SW1 is not necessary and does not need to be assembled. If a reset was necessary, we usually did it via power cycling.

The connector J3 is intended for the 5V supply of the sensor and is parallel to the USB supply (in case the USB supply is not sufficient). However, we have always used the USB supply.

The status of the Hall sensors is displayed via LEDs D1 to D7. There are 8 sensors connected, but the microcontroller only has 7 pins left in the end. Therefore, sensors 7 and 8 share one LED.

The Hall sensors are read out via I2C. A high baud rate is therefore necessary for the fastest possible communication, which is limited by the performance of the I2C drivers in the Hall sensor and microcontroller and by the pull-up resistors R7/R11. The calculation of the pull-up resistors is explained in the schematics and can be adapted by you if you want to connect a different number of Hall sensors, for example. In our setup, 100kBaud has been successfully and reliably tested.

2.1.2 Addressing the IC's

The used Hall sensors can be set to 16 different I2C slave addresses by hardware (external voltage at pin Adr0 and Adr1) (see data sheet, page 22).
This external voltage is generated via two voltage dividers

R_Ad0_HS || R_Ad0_LS
&
R_Ad1_HS|| R_Ad1_LS

Hall sensor	I2C address	Ad1			Ad0		
		U	*R_HS*	*R_LS*	*U*	*R_HS*	*R_LS*
H1	0x96	0 V	nc	0	0 V	nc	0
H2	0x99	0 V	nc	0	3.3 V	0	nc
H3	0x100	1 V	23.2k	11.5k	0 V	nc	0
H4	0x103	1 V	23.2k	11.5k	3.3 V	0	nc
H5	0x104	2.2 V	11.5k	23.2k	0 V	nc	0
H6	0x107	2.2 V	11.5k	23.2k	3.3 V	0	nc
H7	0x108	3.3 V	0	nc	0 V	nc	0
H8	0x110	3.3 V	0	nc	2.2 V	11.5k	23.2k

nc = not connected/open

To simplify the PCB assembly, the necessary resistors are encoded on the layout:

- **x:** Not connected/open
- **0:** 0 Ohm
- **1:** 11.5 kOhm
- **2:** 23.2 kOhm

2.1.3 Connectors

Phoenix Contact Pluggable Terminal Blocks are installed on the Base Station (see part list). The advantage of these plugs are the screw able connections to the cable, so that no expensive crimping tools are required. Any other plug connection is also possible and can be adapted by you according to your wishes (and also available tools in your lab). One pin of the sensor connector (J1) is intended for connecting the cable shield (reduction of EMC). The shield **must not (!)** be connected to GND on the Sensor Head (preventing ground loops)!

Typically, pins of connectors are nickel-plated (and therefore ferromagnetic). The cable is therefore soldered directly to the Sensor Head and strain-relieved with a cable tie (see details in **Fehler! V erweisquelle konnte nicht gefunden werden.**).

2.2 Mechanical Assembly

2.2.1 Sensor Head

After both PCBs of the Sensor Head have been soldered, they should be electrically tested (the solder joints of the Hall sensor IC caused us problems until they worked perfectly). To do this, connect each board directly to the Base Station and test them one after the other (the software recognizes when a sensor is not accessible - in this case the sensors on the other board - and skips the value).

The assembled sensor is shown in Figure 3 is shown.
The connecting cable is soldered on directly. We have used a shielded cable with a length of 90 cm (so that the Base Station is outside the magnetic field). A longer cable is also possible. However, in case of longer cables, the baud rate of the communication may be reduced due the additional capacity between the I2C lines and must be adjusted in the software.
We have used a shielded 5x0.14mm² cable, whereby one wire has not been used. The used cable is called: Lapp Kabel Unitronic LiYCY, 5x0.14, article number 0034305.
The shield is only connected to the Base Station to prevent ground loops.

The distance between the sensors on both boards should be 10mm. Our PCBs have a thickness of 1.6mm, so the printed spacers have a height of 8.4mm (10mm-1.6mm). The CAD of the spacer is located in:

...\1_HallSensor\11_Hardware\112_CAD Files\SensorHead

The hole in the center of these spacers (diameter 2.5mm) is the core hole of an M3 thread, which was cut after printing.

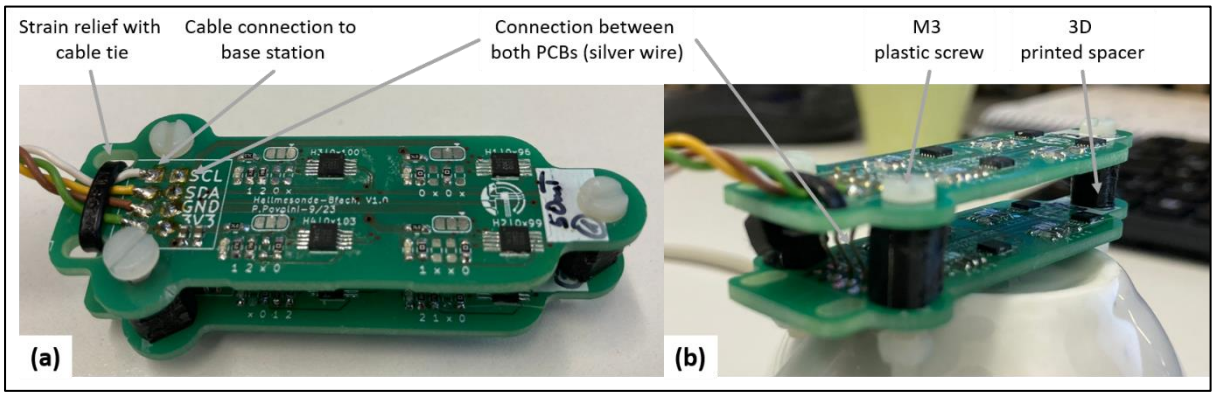


Figure 3: Structure of the Sensor Head

The following procedure made sense for the mechanical assembly:

1. 3D printing of the spacers
2. Screw the 2 boards together.
3. Soldering silver wire (connection between the two PCBs). The front pads of the connector are intended for this purpose.
4. Soldering the cable to the top PCB
5. Attach cable ties for strain relief

2.2.2 Base Station

For us, it was enough to protect the Base Station with 2 plastic covers (against short circuits when something is placed on it, as well as simple ESD protection), see Figure 4.

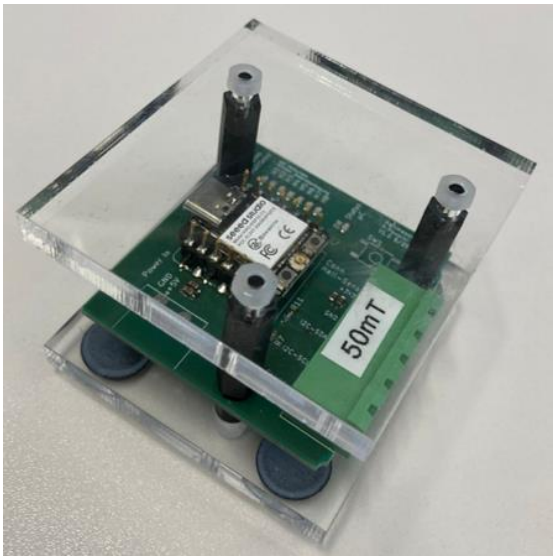


Figure 4: Base Station with 2 Plexiglas covers. Here 4mm thick.

The CAD of the covers is located in:

…\1_HallSensor\11_Hardware\112_CAD Files\Base

In our case, we laser-cut the covers. However, it can also be 3D printed or the sensor can be installed in a proper casing.

3 Embedded software (C++ code using Arduino IDE)

3.1 General

Using the Arduino IDE is a very good way to program the ESP32 microcontroller as easily as possible. The source code (C++, extension .ino) is located at:

...\1_HallSensor\12_Software\ArduinoCode_Seeed_XIAO_ESP32_50mT

The source code is commented in detail; therefore, we forego a detailed description here.

The most important parameters that you can use & adapt are the following in the file "ArduinoCode_Seeed_XIAO_ESP32_50mT.ino":

- const long i2cclock = 100000;
  Baud rate of the I2C protocol. If you have difficulties with the communication between the Hall sensors and the microcontroller, you can reduce the baud rate here (e.g. to 65k).
  The following applies for I2C: Baud rate = bus frequency
- const float Hall_Sensitivity = Hall_Sensitivity_500;
  The software **does not** automatically recognize whether a 50mT, 100mT or 200mT Hall sensor is connected. You can set this here.
  50mT variant: Hall_Sensitivity = Hall_Sensitivity_500;
  100mT variant: Hall_Sensitivity = Hall_Sensitivity_1000;

200mT variant: Hall_Sensitivity = Hall_Sensitivity_2000 = 1.0 (currently Hall_Sensitivity_2000 is not defined)

- const int Repeat_Steps = 1000
  Number of internal averages of the measured value of all 8 sensors
- const long baud_rate = 115200;
  Baud rate of the serial connection with your PC. We wouldn't change it, but maybe it's necessary for you if your PC software can't keep up

After a successful measurement, the data are built to the following string, which is transmitted serially in a single line (here without an external field):

```
H1;Bx=-0.066700(-1);By=0.030600(-6);Bz=0.757450(29);Temp=23.759716;ND=1000
|H2;Bx=-0.197575(-11);By=-0.156925(-6);Bz=0.454975(18);Temp=23.677210;ND=1000
|H3;Bx=-0.070525(15);By=0.232950(13);Bz=0.416950(17);Temp=23.482048;ND=1000
|H4;Bx=-0.048675(-16);By=0.013850(-14);Bz=0.297500(12);Temp=23.686794;ND=1000
|H5;Bx=0.008625(-2);By=0.019700(14);Bz=0.199800(7);Temp=23.779552;ND=1000
|H6;Bx=-0.144675(-5);By=0.125450(1);Bz=0.418250(20);Temp=23.450268;ND=1000
|H7;Bx=-0.098025(-6);By=-0.017600(-3);Bz=0.279500(13);Temp=23.851807;ND=1000
|H8;Bx=-0.113375(9);By=0.157375(7);Bz=0.764075(29);Temp=23.389668;ND=1000
```

The values directly after "Bx", "By" and "Bz" correspond to the measured value in mT. The last raw value (used for debugging) follows in brackets. The temperature (in °C) is used for calibration (Hall effect is temperature-dependent). ND (newData) indicates how many data points could be successfully transmitted and averaged. This value should correspond to "Repeat_Steps". If not, the I2C bus speed should be reduced.

3.2 Installation ESP32 Environment

A new library must be installed in the Boards Manager so that ESP32 chips can be programmed and flashed using the Arduino IDE:
You can find official instructions here:
https://support.arduino.cc/hc/en-us/articles/360016119519-Add-boards-to-Arduino-IDE

We have used the following library:

esp32, V3.0.7 by Espressif Systems

3.3 Flash

To successfully flash the microcontroller, the correct board and port must be selected.
You can find the port (in Windows) using the Device Manager under Ports (COM & LPT), e.g. “COM10”.
You must select the following board: "XIAO_ESP32C3"

Flashing takes about 1 minute, whereby the project is built and compiled beforehand.

4 Calibration

The Hall sensors have small tolerances between each other due to the manufacturing process. As we want to use all 8 sensors to measure the magnetic field, the deviations between the sensors should be as small as possible.
Calibration makes it possible to align the sensors and thus achieve a better measurement result. If only a single sensor were used, this would not be necessary, as the relative accuracy between 2 measurements is really good for modern Hall sensors (however, NMR-based sensors are senseful for absolute accuracy).

A cylindrical coil was built for the calibration, as the magnetic field can also be precisely determined by a precise current measurement. The cylindrical coil has 2 layers: A continuous winding (1st layer) generates the main magnetic field. An additional winding at both ends (2nd layer) homogenizes the magnetic field in the center.

However, please note that calibration is not absolutely necessary! If calibration is not possible for you, then you can skip the following steps.
Avoiding calibration increases the error in the magnetic field mapping, but this may be compensated for by the interpolation between the measuring points implemented in the analysis of the measurements. However, the result has not been evaluated by us and should therefore be checked.

4.1 Software to calculate magnetic field

4.1.1 Analytical Solution using Biot Savart

- *General*

The analytical calculation was written in Matlab and is available at:

...\2_Calibration\21_Hardware\211_CalibrationSolenoid_Analytical

The magnetic field is calculated using the analytical formula according to Simpson et al (https://ntrs.nasa.gov/api/citations/20010038494/downloads/20010038494.pdf) by placing round conductor loops parallel to each other.

The software for calculating the magnetic field is being run iteratively. The following procedure made sense to us. This could be automated in a further development:

1. With "Main_CalculateSolenoid.m" the magnetic field in the center of the cylinder is calculated, depending on the parameters you have set. At this point we don't know the optimal number of turns of the 2nd layer and we estimate it (e.g. 25). We try to determine the optimal cylinder geometry (diameter, length) and the copper wire diameter with which you can achieve the desired field strength at the desired current (limited e.g. by available current sources)
2. In the "getOptimumSecondLayer.m" script, the parameters (diameter/length of cylinder, copper wire) must have the same values as in 1). The script then calculates the optimum number of turns of the 2nd layer.

3. Insert the calculated number of turns back into 1) and check whether the calculated field strength and homogeneity are correct.

The software is intensively commented, which should make adaptation possible.

We used a DN160 KG pipe (radius 80mm) as the cylinder core. 290 windings were on layer 1. The insulated copper wire has a diameter of 1.5mm, plus 30µm enamel thickness. This results in a cylinder length of 452.4mm. The 2nd layer has 27 turns.

4.1.2 FEM Simulation using free Software FEMM & Python

The analytical simulation is very fast, but only assumes the conductor loop as an ideal path. This error can be checked again in a FEM simulation. The open source tool FEMM V4.2 with the Python interface pyFEMM is used: https://www.femm.info/wiki/pyFEMM
The Python code is located at:

...\2_Calibration\21_Hardware\212_CalibrationSolenoid_FEM

The software is commented in detail so that it can be easily customized. Essentially, the Python script calls the FEMM software in the background, sets up the geometry, adds the material parameters and the flowing current, starts the FEM simulation, reads out the result and saves it in a CSV table (for further processing).

We have found that pyFEMM can become unstable if too many windings are built up in FEMM. Therefore, the entire simulation was split into 8 individual FEM simulations and the results were super positioned at the end.

4.2 Structure of the hardware

The CAD of the cylinder coil is in:

...\2_Calibration\21_Hardware\214_CalibrationSolenoid_CAD_STEP

the assembly is divided into 2 parts:

- Cylinder coil (water-cooled)
- Slider (mounting of the Sensor Head)
- Current measurement setup for calibration

and is described below

4.2.1 Cylinder

The cylinder is shown in Figure 5.

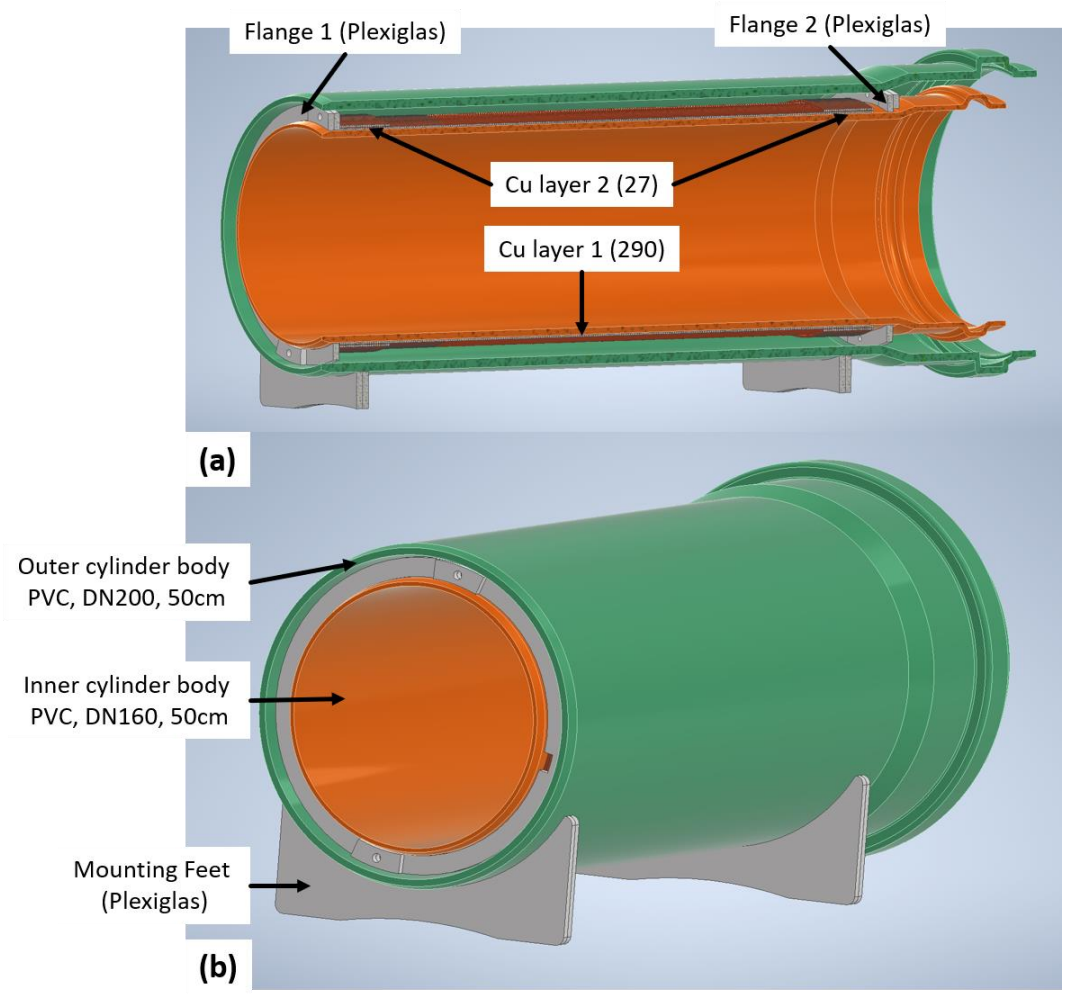


Figure 5: CAD rendering of the assembled cylindrical coil

- *Inner Cylinder*

An ordinary waste water pipe is used. We bought this pipe:
Ostendorf Kunststoffe, No. 222000, KG EM Rohr SN 4 Coex DN/OD 160 x 500 mm
https://www.ostendorf-kunststoffe.com/en/products/waste-water-pipes/the-kg-system-pvc/

- *Outer cylinder*

The electric coil becomes hot very quickly due to losses. It is therefore water-cooled (as heat capacity). The coil is sealed with a second waste water pipe:
Ostendorf Kunststoffe, No. 223000, KG EM Rohr SN 4 Coex DN/OD 200 x 500 mm
https://www.ostendorf-kunststoffe.com/en/products/waste-water-pipes/the-kg-system-pvc/

To be able to seal flange 2 properly, the end is shortened slightly with a saw (Figure 6). A small hole (1 mm) was drilled into the top of this cylinder to release the pressure from the cylinder when the temperature increases (and thus the volume expands).

Figure 6: Photo of the coil

- *Flange 1 (Front)*

The flange consists of 3 plastic parts, see Figure 7. In our case, everything is laser-cut from 5mm thick Plexiglas. But the parts can also be 3D printed or milled from other materials.

Two holes are provided here for the water inlet and outlet. An M6 thread is cut into these after lasering. During the measurement, the volume is sealed with a plastic screw, a washer and an O-ring (under the washer) (Figure 7c). The Reinforce part was fully glued to the base. The adhesive used was 2K epoxy Loctite EA 9466 (RS-Online 244-2026) (slightly yellow in the photo), as the adhesive stays a bit elastic when cured and is therefore better able to withstand temperature fluctuations. The same adhesive was used to glue the flange to the cylinders over the entire surface. **The PVC of the cylinders must be properly sanded, otherwise the glue joint can hardly withstand any load**.

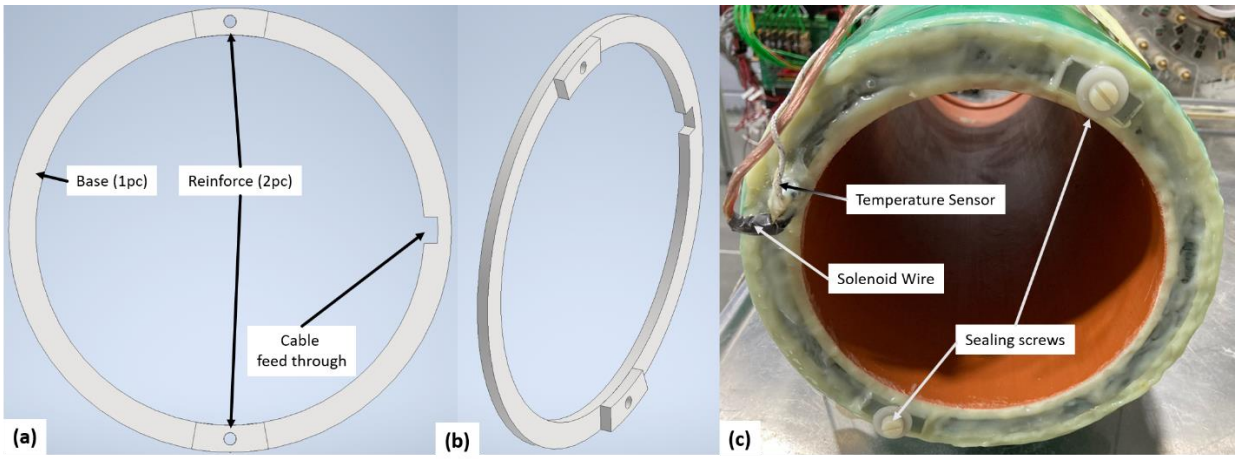


Figure 7: Flange 1 (Front)

- *Flange 2 (Back)*

The rear flange has a slightly different design (see Figure 8). We broke this part when winding the cylinder coil (see details later), so it is designed to be "replaceable" (as it can't be pulled over the thick flange of the waste water pipe) in case it breaks during winding and needs to be replaced.
The base is divided into 2 parts (can be placed around the cylinder afterwards). The 2 halves are connected (not glued) using the Reinforce parts. They are screwed together with M4 plastic screws (length 16mm). Once everything is assembled and the cylinder needs to be sealed, everything is sealed with plenty of glue.
All parts are laser-cut from 5mm plexiglass. Any other manufacturing method is possible too.

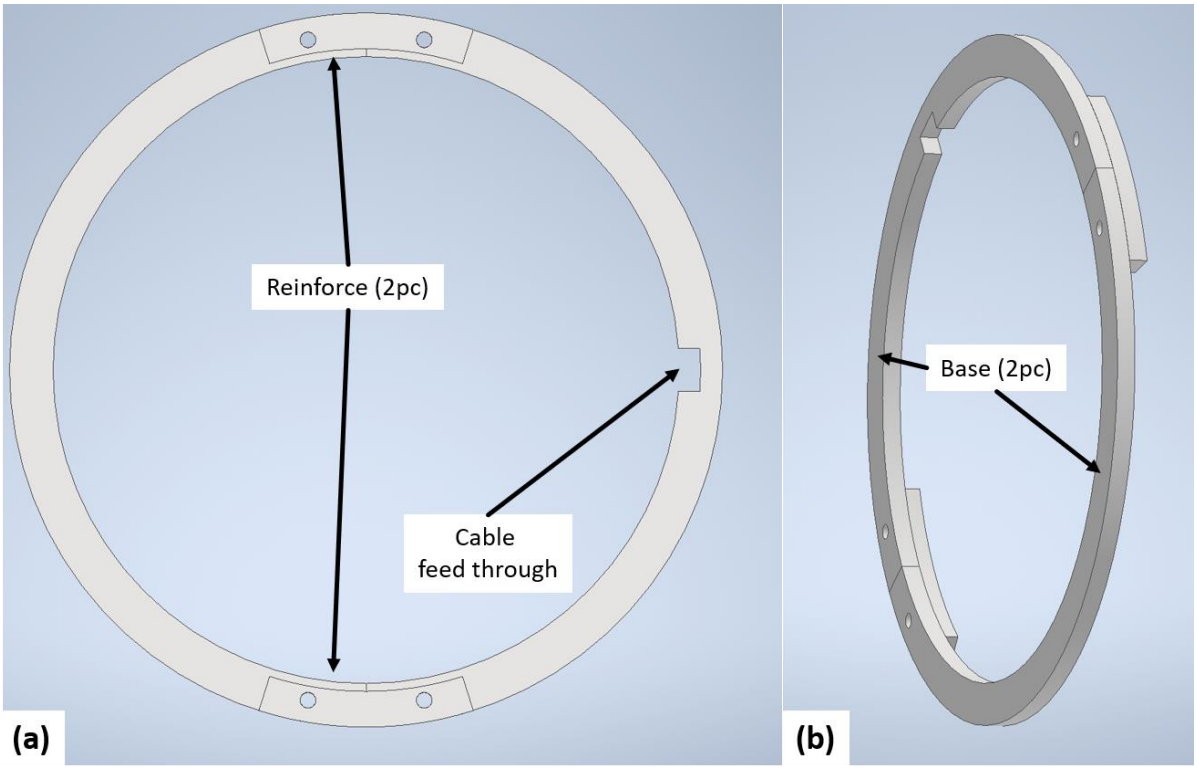


Figure 8Flange 2 (Back)

- *Mounting Feet*

In our design, each foot consists of 2 pieces of 5mm thick Plexiglas sheets (for a larger gluing surface with the sanded PVC cylinder).

- *Cu Layer 1*

We used a large lathe to wind the coil. The end without the flange of the water pipe was clamped in the chuck. An original DN160 cover cap was installed in the flange itself (Ostendorf Kunststoffe, No: 222620, KGM Muffenstopfen DN/OD 160), in which the center point of the lathe was inserted.

Flange 1 was mounted directly at the beginning (in the direction of the chuck) to enable a clean stop for the windings (see Figure 9).

The coil was turned at the slowest speed of the lathe, keeping the copper wire under tension and ensuring that the windings were as tight as possible. After reaching the correct number of windings, flange 2 was mounted and the wire was fed through the feed through and cut off.
A temperature sensor (type K, RS-Online 136-5868) was glued to the coil in order to detect an excess temperature during operation.

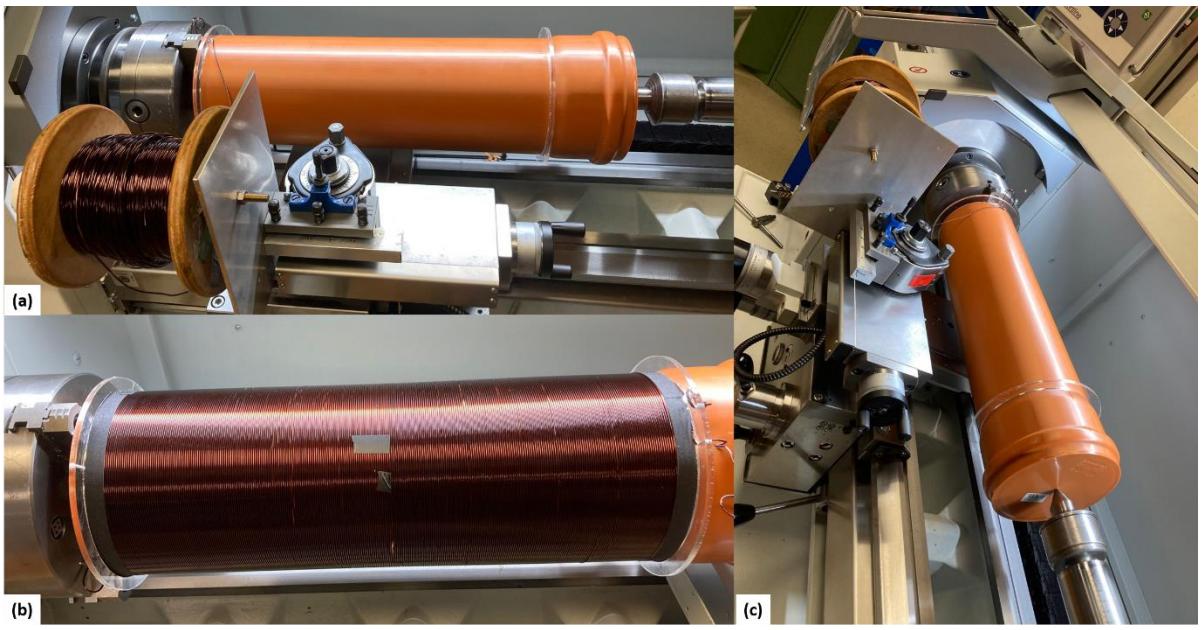


Figure 9Winding layer 1 on the lathe

- *Cu Layer 2*

Layer 2 is wound in the same way as layer 1. Make sure that the winding direction for layer 1 is retained. The 3 windings are then interconnected (see diagram in Figure 10).

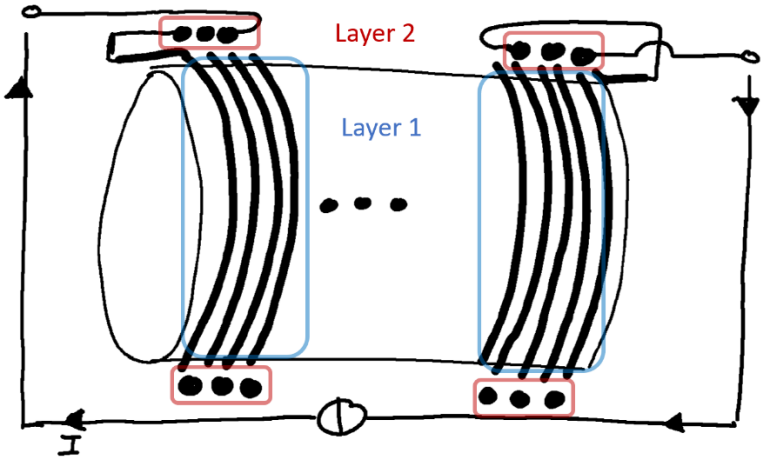


Figure 10Interconnection of layer 1 and layer 2

After both layers were wound up, the electric coil was wrapped with adhesive tape and secured in this way.

4.2.2 Slider

The slider (see Figure 11a) is inside the solenoid and holds the Sensor Head in the isocenter of the solenoid. To measure all 3 axes individually, the sensor can be mounted in different positions (Figure 11b). The slider consists of many parts and the assembly is explained below.

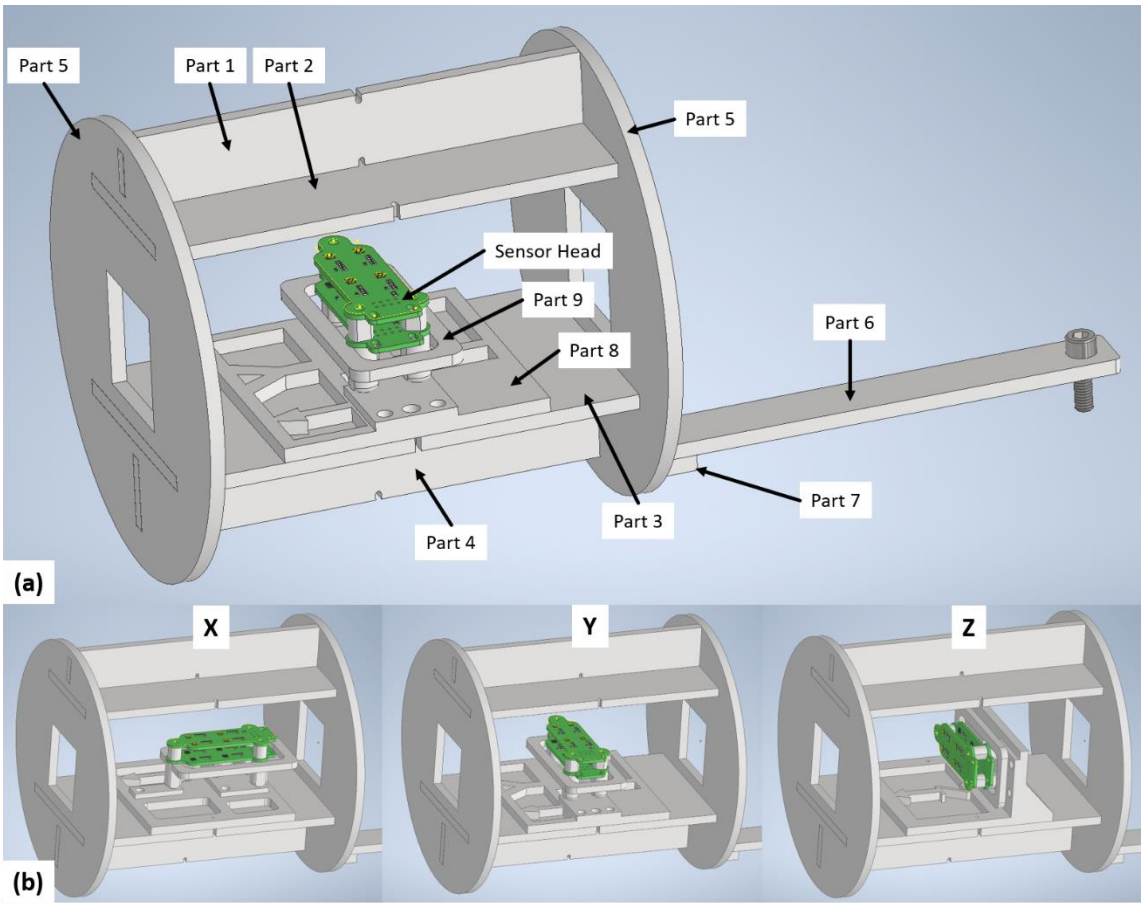


Figure 11(1) CAD Picture of used slider. (b) Mounting of Sensor to measure different axes

- *Part 1-5*

The main body of the slider is again made of laser-cut 5mm thick Plexiglas. These 6 parts (2 times Part5 is used) were assembled with superglue. Any other glue will also work

- *Part 6*

Plexiglas, 5mm thick, laser-cut. This part is the spacer that holds the slider in the isocenter (using the plastic screw that rests on the edge of the cylinder).

- *Part 7*

3d printed angle. Glued to Part 5. Part 6 and Part 7 are connected via a bolt (screw)

- *Part 8 and 9*

3D printed holder for the Sensor Head, with which each of the 3 axes can be individually calibrated for the 8 Hall sensors. Hall sensors 1 and 2, H3 and H4, H5 and H6, H7 and H8 were measured simultaneously, as the homogeneity of the cylinder is good enough.

Part 8 is centered with Part 3 via M2 screws (in the holes). Additional rubber bands have firmly connected the parts together.

The Sensor Head is screwed to Part 9 (M3 thread cut into Part 9). This assembly is again attached to Part 8 using rubber bands. Self-adhesive cable clamps were attached to the slider to have attach points for the rubber bands (RS-Online 184-2343).

4.2.3 Current measurement and operation of the cylinder coil

The setup of the current measurement is also described in the SI of the paper. Therefore, only a brief scheme in Figure 12. Figure 13 shows pictures of the measurement.

The setup depends heavily on the hardware you have available (shunt, relay, multimeter). The shunt should be sufficiently well cooled so that the resistance does not drift due to temperature changes. The relay can also be replaced by a powerful switch, or is unnecessary if a sufficiently powerful power source is available. A "normal" relay is also possible (we only had a solid state relay left).

Make sure that the multimeter is sufficiently accurate in the expected voltage range (<1V). A communication interface is advantageous so that calibration can be automated. With the one we used (GW-Instek GDM-8255A), this was possible via USB and serial interface. For powers up to 30A/60V we had 3 lab voltage sources (EA-PS 3065-10B) in parallel operation (indicated in Figure 12). For higher currents, we used 8pc 12V-batteries connected in series (at charging end voltage 110V/69A), see Figure 13.

If you also want to use the battery solution, make sure to prevent the batteries from short-circuiting! A DC electric arc is difficult to switch off again!

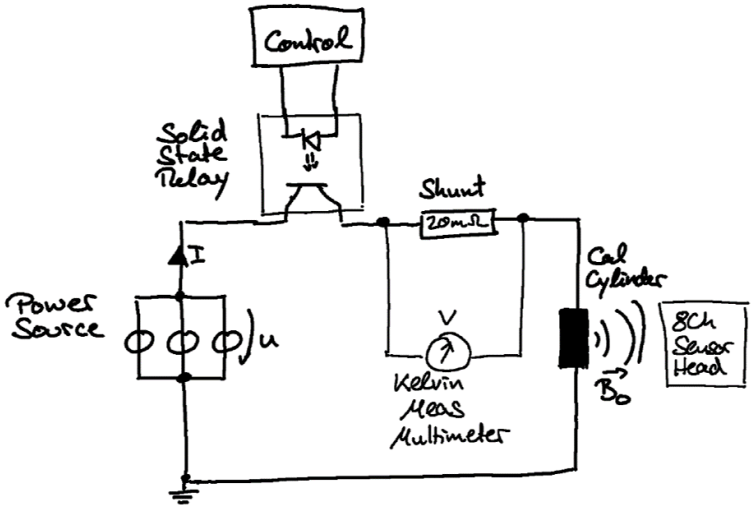


Figure 12: Schematic diagram of the current measurement

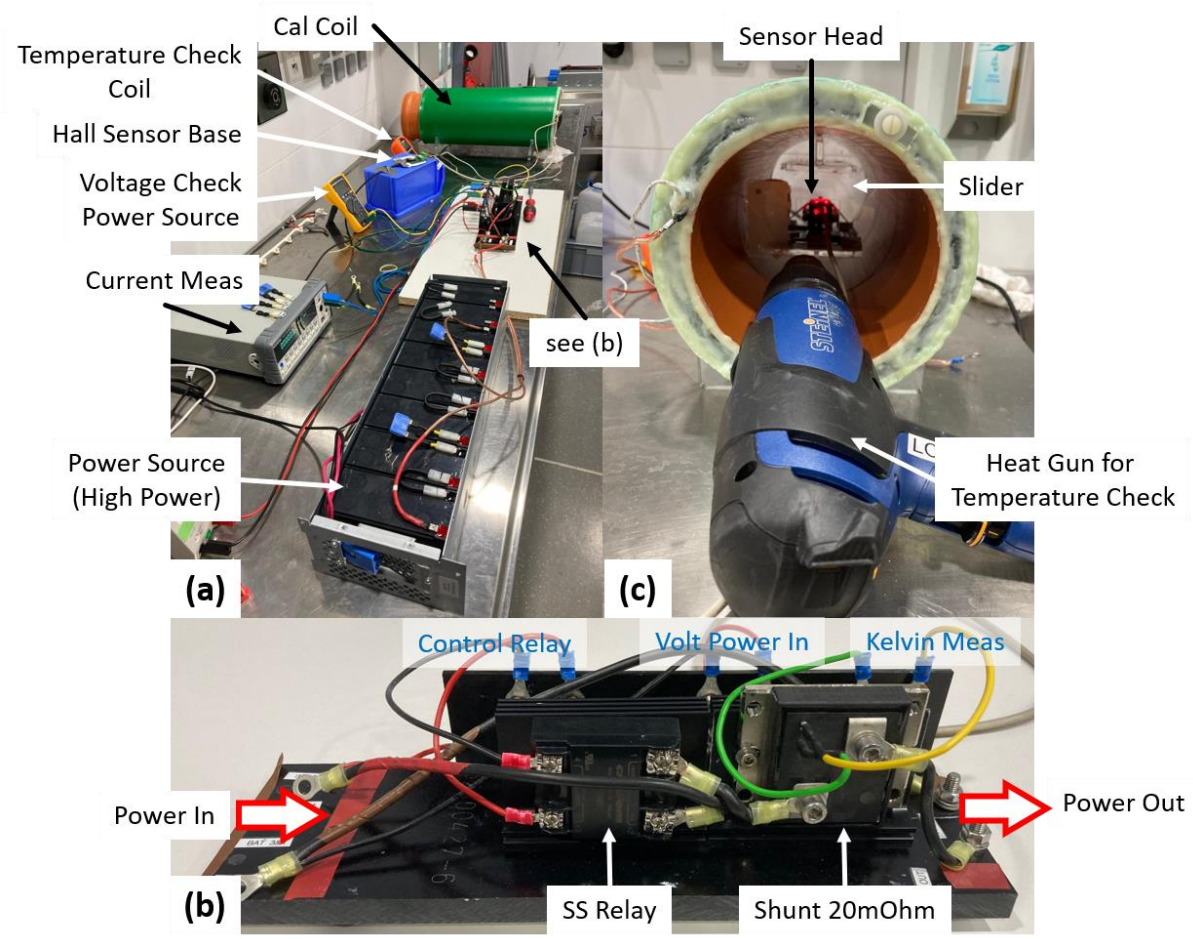


Figure 13: (a) Operation of the coil during a calibration sequence. In this case, the 8 serial 12V batteries generate the current flow. (b) Set-up of the current measurement. (c) Sensor Head in the cylinder. Hot air gun for temperature change of the Sensor Head.

- *Alternative current measurement*

Multimeters are probably available in every electrical laboratory. However, access to a multimeter with a communication interface can be difficult. In particular, we had difficulty finding a multimeter with which communication was "easily" possible (without the need for special/expensive toolboxes, licenses, etc). We were lucky that with the GW-Instek we happened to already have one that could be read out serially.

If you have major difficulties at this point, the following solution could be useful. The setup is shown schematically in Figure 14.

Based on an Arduino (no matter which model - for example the Arduino Uno, which is used in the robot controller) and an external 16bit ADC, you can build a precise system:

The external ADC ADS1115 from Adafruit Adafruit (https://www.adafruit.com/product/1085#technical-details, Mouser Number: 485-1085) is an I2C adjustable ADC with 16 bit and adjustable gain. This is a suitable library for the Arduino IDE: https://github.com/adafruit/Adafruit_ADS1X15

You can find more details on setting up the ADC in this tutorial:

https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/

You can use the "continous" sample script:

https://github.com/adafruit/Adafruit_ADS1X15/blob/master/examples/continuous/continuous.ino

As far as we can tell (we have not used the system), you hardly need to change anything in this script. What we would recommend:

- Line 23: Changes the serial baud rate to 115200 Bd
- Line 24/26/27: Serial texts for start could be deleted (optional)
- Line 38: Remove comment / activate "ads.setGain(GAIN_FOUR);"
- Line 64: Clean up serial text. Replace this line with:
  "Serial.println(ads.computeVolts(results));"
  Then the serial output of this system corresponds to the multimeter we are using and you hardly have to change anything in the following scripts.
- Line 70: delete delay(1000). A hardware interrupt is used so that a serial string is sent out with each new ADC sample. The sample rate is 860Sps, so you get a new serial measurement every ~1.2ms

This accuracy results in a maximum precision of 1bit = 0.03125mV. Depending on the shunt used, you can calculate your maximum resolution. In our case, this results in a current resolution of $\Delta I = \Delta V / R = 1.5mA$ or, with the calibration coil used, a resolution of the magnetic field of approx. 1.2µT. Of course, additional noise is to be expected.

This ADC board costs approx. 14€. If an already available Arduino is used (e.g. the Uno from the robot controller, or a cheap Nano etc), a very inexpensive & integrated system can be set up.It makes sense to check the performance with an industrial multimeter.

A separate PCB is actually not required for this setup. For reliable operation, however, the PCBs should be installed in a casing (i.e. to strain relief the cables).

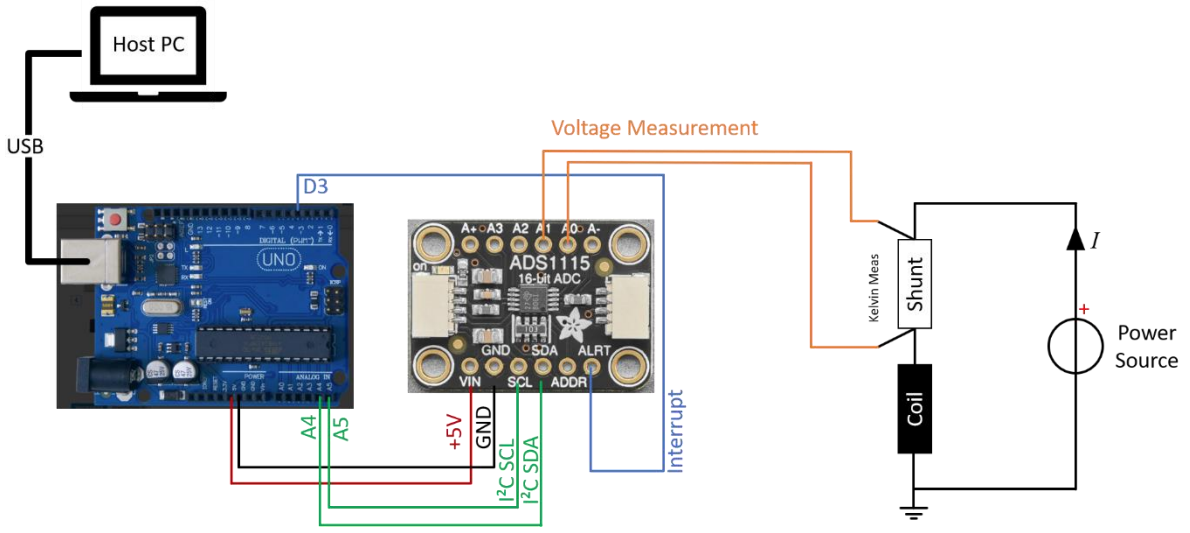


Figure 14: Possible setup of your own "multimeter" using an Arduino UNO and the Adafruit board (16bit ADC ADS1115)

4.3 Calibration Run

4.3.1 Procedure

In our case, 3 experiments were carried out to calibrate the Hall sensors:

1. Linearity in a varying magnetic field (Ch. 4.3.2)
2. Temperature dependence in the static magnetic field (Ch. 4.3.3)
3. Static offset (Chap. Ch. 4.3.4)

4.3.2 Calibration Run: Linearity

- *Position/Rotation Sensor Head*

Each axis for each sensor is calibrated individually, whereby Hall sensors 1 and 2, H3 and H4, H5 and H6, H7 and H8 are each measured together (see example for x-axis in Figure 15). This means that 8 measurements are taken per axis (a total of 3x8 = 24 individual measurements).

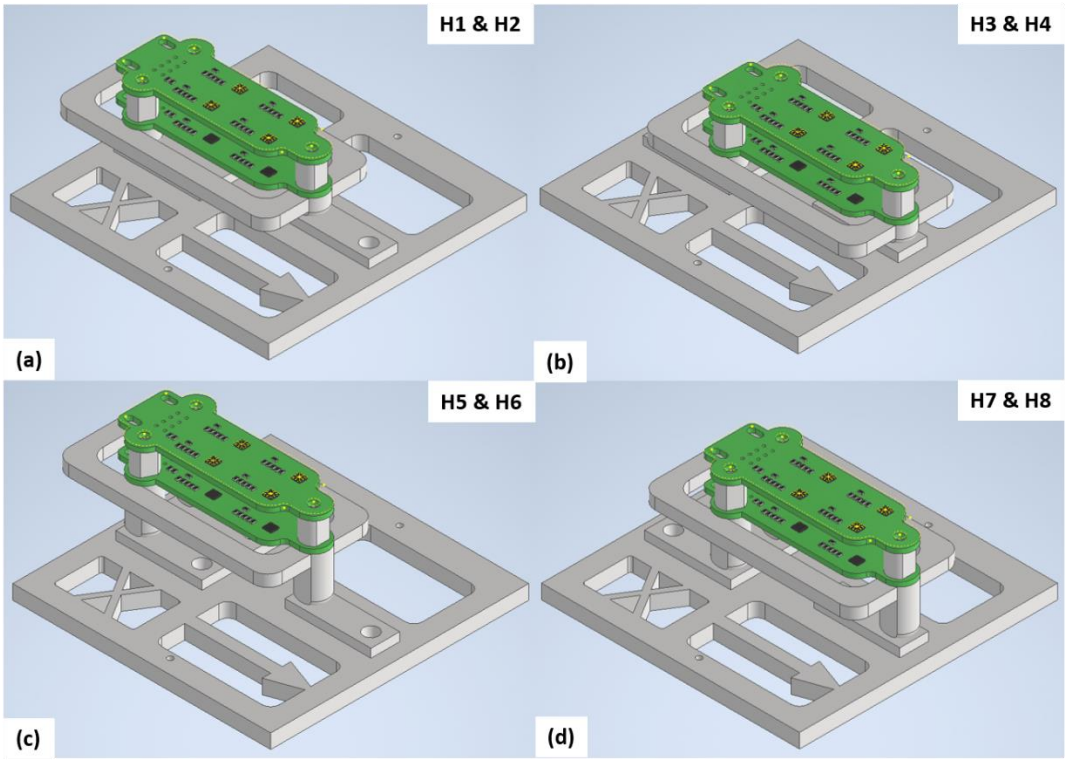


Figure 15: Example of positioning the Sensor Head for calibrating the X-axis. (a) Hall sensor H1 and H2 are in the isocenter. (b) ... (d) All 4 positions for calibration.

The feet of the Hall sensor holder are self-centering. This means that the user must ensure that the Sensor Head rests firmly (and is not jammed in any way). The magnetic field of the cylinder is rotationally symmetrical, so the rotation of the slider is not critical.
However, it is important to note that the measurements are carried out in a normal laboratory room. There are therefore influences from other devices/earth magnetic field etc. Therefore, the calibration cylinder should be in the same position during all measurements (we used an edge in the table for this and aligned the cylinder to it).

- *Used Software*

The Matlab software for carrying out the linearity measurement can be found here:

...\2_Calibration\22_Software\221_Matlab_ReadOut_DuringCalibration

The script "Main_Calibration_Linearity.m" is used. The software is commented in detail, hence the reference to the source code.

The most important changes before your experiments are:

Depending on your multimeter, you will need to change the type of communication and how the measured values are tapped. To do this, adapt the "FunctionGetCurrent" function so that the "Volt_Meas" variable gives the measured voltage in volts. This is accompanied by an adaptation of the "% SetUp Communication with Multimeter" block in the main program

Immediately before the experiment (setup the script):

Adjusts the LogFile and all SetUp parameters for documentation (e.g. Multi_R_Shunt, Shunt_Type, HallSensor_Type etc).

During experiment:

Parameters:

- num_AQ: Number of measured points at the same magnetic field strength (for more precise calibration)
- num_BField_Steps: Number of measured magnetic fields per axis & sensor
- measured_Sensor: Specifies which sensor is currently being measured (as string "H1"). In each case 2 sensors are measured simultaneously. Enter the smaller index here
  (e.g. for H1 and H2 -> 'H1')
- measurend_Direction: Specification of the momentary measured axis as a string ("x", "y", "z")

Start the script. It establishes the connection to the Hall sensor and multimeter. And then immediately evaluates the first measuring point. Once the measurement is complete, this is indicated in the command line. The measurement points are stored temporarily in the "LogFile" struct. After the experiment, the file is saved to the hard disk.

- *Current change*

We took 26 measuring points per calibration run. The 3 parallel voltage sources (in CC mode) were used for currents up to ±30A. For higher currents, the battery bank was used (once connected as 7seriell and once as 8seriell).

The software signals when the current can be changed. The current flowing is then changed manually (adjusting the power sources). As soon as the current flows, this is reported to the still running Matlab script (push “Enter” in the command window) and the measuremnt is done at the next magnetic field. Of course, this can be automated in a subsequent development.

A lot of ohmic power loss occurs during the measurement and heats up the copper considerably, especially at high power levels (our coil has approx. 1.6 Ohm -> 40A leads to 2.5kW). Using the temperature sensor, make sure that the coil temperature does not exceed 60°C (approx. value,

protects the copper insulation and the plastic used). Care should be taken to ensure that the Hall sensor does not become too hot.
If it exceeds this temperature: Wait until it has cooled down (takes a long time). Or replace the cooling water with cold water via the built-in threads in the front panel.

4.3.3 Calibration Run: Temperature Dependence

The Hall effect is highly temperature-dependent. A large part of this error is already corrected by the integrated electronics in the Hall sensor IC itself. However, the Hall sensor heats up due to the losses in the cylindrical coil. Therefore this experiment.
The script "Main_Calibration_TemperatureDep.m" is used. In principle, there is no difference to the instructions in the script in 4.3.2. The only difference is that several iterations are not run through and the parameter "num_AQ" is significantly larger (e.g. 100). Set a constant magnetic field (in our case 20mT) and start the measurement.
During the experiment, we raised the temperature to 40°C using a hot air gun (see Figure 13c).

This measurement is carried out simultaneously for all Hall sensors per axis. The associated error (due to inhomogeneity) is compensated for in the calibration calculation by a weighting factor. During the measurement, mount the Sensor Head in the position of Hall sensor H1 and H2 for all 3 axes.

4.3.4 Calibration Run: Static Offset

This step only makes sense if you have access to a Zerofield Chamber. In our case we had access to one, where other experiment are carried out. Or you have access to a small one, like the ones typically used for other magnetic field sensors (like this one).

The "Main_Calibration_ZeroOffset.m" script is used for the measurement. Here you only have to adjust the COM_Port for your Hall sensor. The rest is done automatically.

4.4 Evaluate Calibration & get Calibration Parameter

The scripts for the evaluation are available at:

...\2_Calibration\22_Software\222_Matlab_Calibration

Copy all logged measurement data from the calibration into the "Log_Folder" folder. All measurement files from this folder are evaluated together in the script.
The magnetic field of the calibration cylinder is not perfect due to geometric errors. This is shown for example by the fact that the calculated calibration factors of the sensors are outside the data sheet specification. Therefore, the static error of the calibration cylinder is first determined and minimized using the data sheet specification. Use the script "Main_PostProcessing_Calibration_calcCalError.m" for this.

Nothing needs to be changed by you. The functionality of the software is intensively commented. The evaluation takes a while (running through many fmincon optimizations).

At the end, 3 figures are created, whereby plot 2 is used to evaluate the error of the calibration coil (see Figure 16). We take advantage of the fact that the Hall sensors are tested by the manufacturer and produced within certain tolerances.

The upper plot shows for each axis how many sensors exceed this manufacturer tolerance for an assumed coil error. We set the first data point at which the x and y axes have an error of 0 as the error of the calibration coil. In our experiment, the z-axis error then increases, which could be explained by the larger manufacturing tolerance at this axis. We have chosen -4.2% as the error.

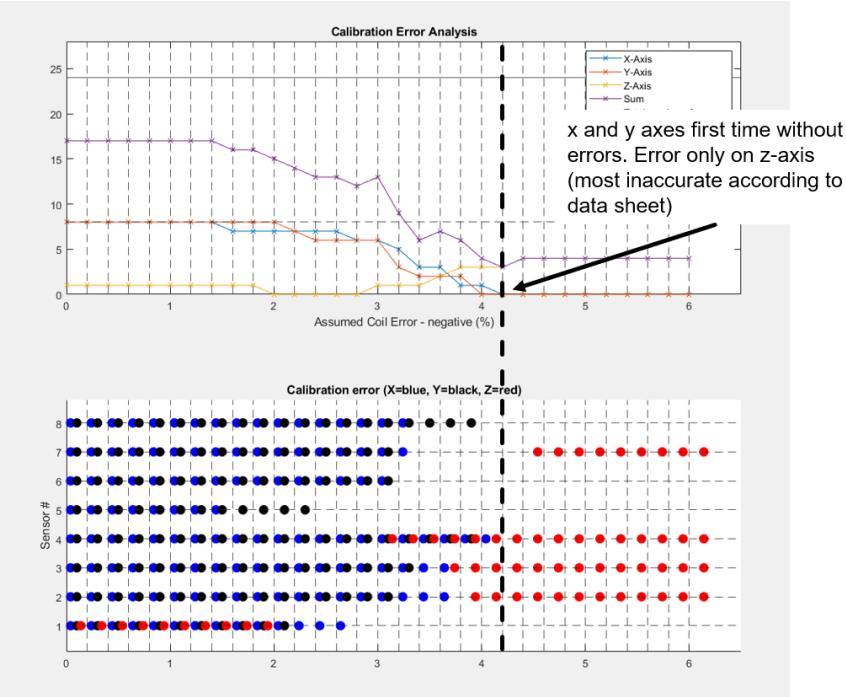


Figure 16: Exemplary result of the error estimation

This value is then transferred to the "Main_PostProcessing_Calibration.m" script as the "error_CoilConversion" parameter.

The script runs automatically. The calculated calibration factors (struct CalFile) are saved in a file at the end (e.g. "Calibration_50mT_24-Jul-2024_19_02_55.mat").