

## Gliederung

<b>1 Einleitung</b>	<b>1</b>
<b>2 Funktionsprinzip des Sensors</b>	<b>2</b>
<b>3 Dokumentation des mechanisches Aufbaus</b>	<b>4</b>
<b>4 Dokumentation der Software</b>	<b>5</b>
<b>5 Validierung und Test</b>	<b>9</b>

## 1 Einleitung

Ziel des Microcontroller Praktikums ist es, einen Sensor in Betrieb zu nehmen und dessen Messdaten auszulesen und darzustellen. Der dafür zur Verfügung gestellte Sensor ist in unserem Fall ein Magnetfeldsensor, der für Kompassanwendungen in Mobilgeräten konzipiert ist. Das naheliegendste wäre es also gewesen, einen Kompass zu entwickeln. Diese Aussicht fanden wir jedoch nicht besonders spannend.

Da wir in der Vergangenheit schon häufiger an elektrischen Schaltungen gearbeitet haben, dachten wir uns, dass eine Smartphone App zur Strommessung sehr praktisch wäre. Mit einem Magnetfeldsensor, wie er im vorliegenden IC verbaut ist, sollte es zumindest theoretisch möglich sein, den Strom durch einen Leiter anhand seines Magnetfeldes zu messen. Folgender Zusammenhang gilt für die magnetische Flussdichte um einen geraden Leiter:

$$B = \mu_0 \cdot \frac{I}{2 \cdot \pi \cdot r} \quad (1)$$

Wenn man den Sensor also in konstantem Abstand zum Leiter hält, besteht ein direkter Zusammenhang zwischen dem gemessenen Magnetfeld  $B$  und dem Strom durch den Leiter  $I$ .

Nach diesen Vorüberlegungen hatten wir es uns zum Ziel gesetzt, mit dem vorliegenden Sensorboard einen kontaktlosen Stromsensor zu entwerfen.

Neben der Arbeit, die Sensordaten auszuwerten und eine entsprechende Logik zu programmieren, ergab sich dadurch zusätzlich die Notwendigkeit eines mechanischen Aufbaus. Im Folgenden wird zunächst das Funktionsprinzip des Sensors erläutert. Anschließend folgt die Dokumentation des mechanischen Aufbaus und der Software. Abschließend wird der fertige Aufbau noch mit einigen Tests validiert und dessen Möglichkeiten aufgezeigt.

## 2 Funktionsprinzip des Sensors

Beim vorliegenden Sensorboard handelt es sich um ein Sensorboard der Firma Adafruit, welches neben einiger Zusatzbeschaltung in erster Linie mit dem Sensor-IC LSM303DLHC von ST-Microelectronics bestückt ist. Hierbei handelt es sich um ein "eCompass Modul", das sowohl die Beschleunigung, als auch das Magnetfeld im dreidimensionalen Raum messen kann. Da im Praktikum nur der Magnetfeldsensor verwendet wird, wird im Folgenden auch nur dessen Funktionsweise beschrieben. Der Hersteller macht im Datenblatt keine genaueren Angaben, nach welchem Funktionsprinzip der Sensor arbeitet, vergleichbare Sensoren nutzen jedoch den anisotropen magnetoresistiven Effekt (AMR).

Dieser Effekt wurde bereits 1857 entdeckt, konnte jedoch erst in den 1960er Jahren technisch genutzt werden. Die ersten Sensoren, die sich den AMR-Effekt zunutze machen, wurden in den 1980er Jahren mit dünnen Schichten ferromagnetischer Materialien realisiert. Grundlegend ist das Funktionsprinzip, dass der spezifische Widerstand von ferromagnetischen Materialien parallel zur Magnetisierung einige Prozent größer ist, als senkrecht dazu. [2]

Der Zusammenhang zwischen Winkel und Widerstand ist in folgender Formel beschrieben:

$$R = R_0 + dR \cdot \cos(2 \cdot \alpha) \quad (2)$$

Hierbei ist  $R_0$  der mittlere Widerstand und  $R$  die maximale Widerstandsänderung. [4] Aus der Formel wird außerdem deutlich, dass mit diesem Zusammenhang zwar die Richtung, nicht aber das Vorzeichen des äußeren Magnetfeldes ermittelt werden kann. Dies ist verständlicherweise für eine Kompassanwendung nicht optimal. Dieses Problem wird durch das Einbringen von dünnen Aluminiumstreifen im  $45^\circ$  Winkel in die Legierung gelöst, den sogenannten "Barber poles" wie in Abbildung 1 zu sehen ist. [1]

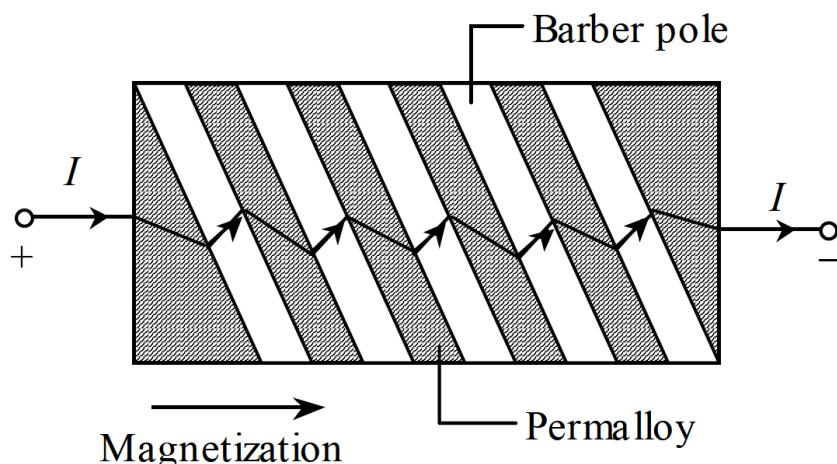


Abbildung 1: Barber poles

Da Aluminium einen deutlich höheren Leitwert als die sonst verwendete Legierung hat, drehen die Barber poles die Stromrichtung um  $45^\circ$ , was den Winkel zwischen der Magnetisierung und dem elektrischen Strom um eben diesen Winkel verschiebt. Vier dieser Elemente werden dann üblicherweise in einer Vollbrücke verschalten, wobei die Barber poles der Diagonalelemente jeweils um  $90^\circ$  zueinander versetzt angeordnet werden. Somit führt eine Erhöhung des Widerstandes im einen Diagonalelement zur Verringerung des Widerstandes im jeweils anderen. Dies ermöglicht es die genaue Richtung des Magnetfeldes zu bestimmen und eliminiert außerdem die Temperaturabhängigkeit der Sensoren und erhöht die Empfindlichkeit der Messung.

Diese Anordnung ist im Sensor dreimal vorhanden und jeweils nach den Koordinatenrichtungen angeordnet, um in jeder Richtung das Magnetfeld messen zu können. Das Prinzip ist auch in Abbildung 2 aus dem Datenblatt des Sensors erkennbar. Hier sind die Messelemente in drei Wheatstone Brücken zu sehen, deren Signale dann über einen Multiplexer und eine Verstärkerschaltung an einen A/D Wandler weitergeleitet und dort in ein digitales Signal umgewandelt werden. Dieses Signal wird dann digital, je nach Konfiguration des Sensors, verarbeitet und über I<sup>2</sup>C versendet.

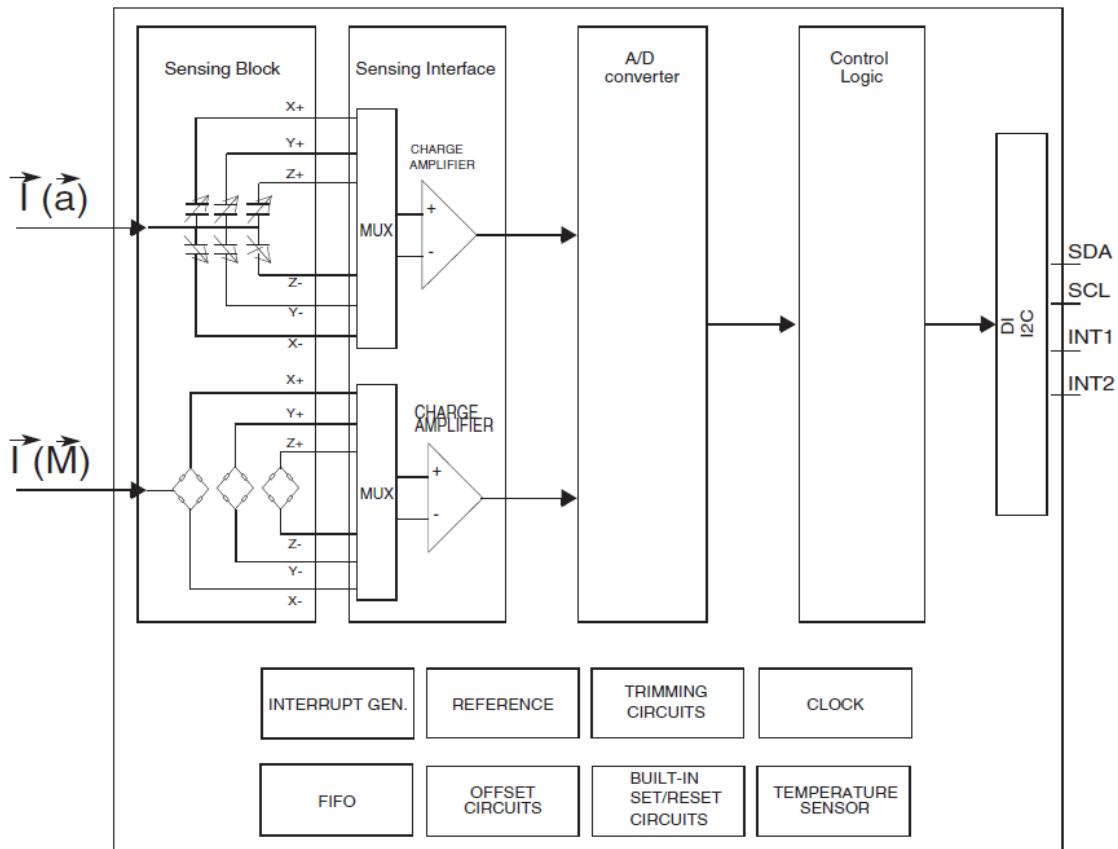


Abbildung 2: Aufbau des Sensors

### 3 Dokumentation des mechanisches Aufbaus

Wie bereits in der Einleitung erläutert, ist eine Messung von absoluten Stromwerten nur möglich, wenn der Leiter immer im selben Abstand zum Sensor gehalten wird. Erste Tests haben gezeigt, dass es bei Leitern unterschiedlichen Querschnittes entscheidend ist, den Mittelpunkt des Leiters im selben Abstand über dem Sensor zu halten. Aus diesem Grund wurde eine Konstruktion entworfen, die eben dazu in der Lage ist.

Zunächst wurde ein detailliertes CAD Modell erstellt. Zwei Renderings daraus sind in Abbildung 3 zu sehen.

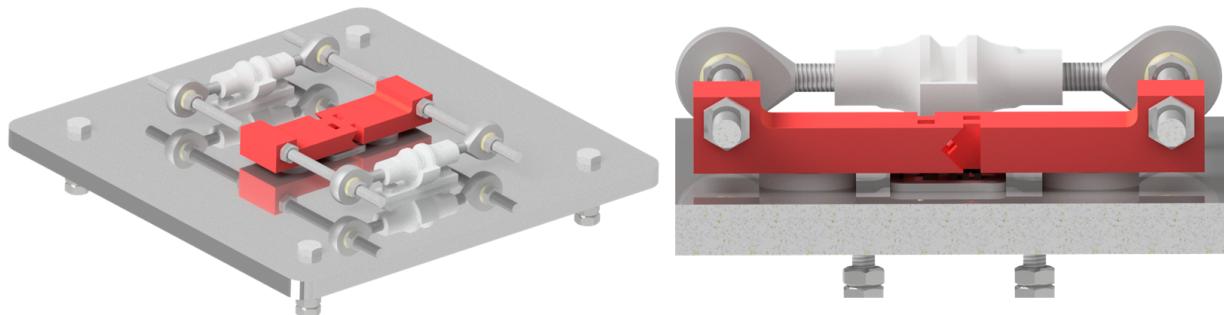


Abbildung 3: Mechanischer Aufbau (Rendering)

Der Sensor sitzt mittig unter dem roten Klemmteil, der Leiter wird von den beiden V-Förmigen Aufnahmen in Position gehalten. Durch Drehen an den beiden weißen Bauteilen können die Klemmteile gleichzeitig aufeinander zu bzw. voneinander weg bewegt werden. Somit liegt der Leiter immer mittig über dem Sensor und unabhängig vom Querschnitt, ist der Abstand konstant. Beides ist für eine vergleichbare Messung unerlässlich.

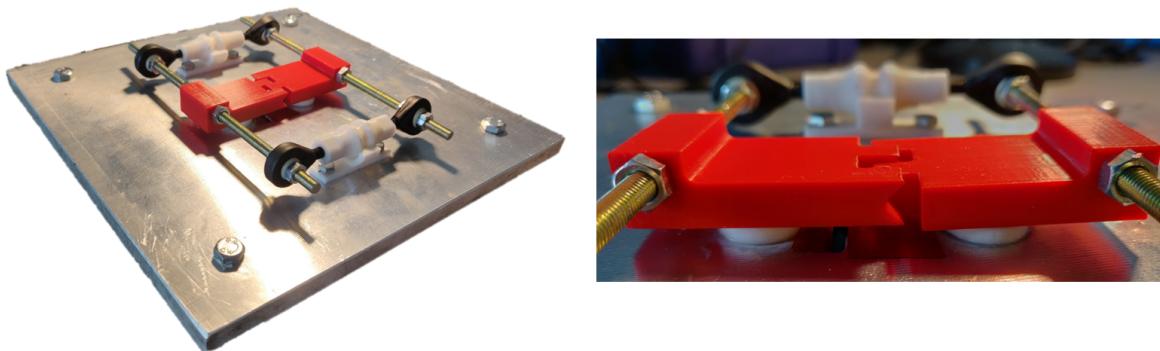


Abbildung 4: Mechanischer Aufbau (Foto)

## 4 Dokumentation der Software

Zur Entwicklung der Software wurde das Meson Build System [6], welches im Hintergrund zum Kompilieren auf Ninja [5] zurückgreift, sowie ein git [3] Repository zur Versionsverwaltung verwendet. Die Kommentierung ist Doxygen [7] konform. Durch die in der Einleitung beschriebenen Ziele und den gegebenen Projektvorgaben ergab sich als Anforderungen an die Software folgende Punkte:

1. Konfiguration des Sensors und Auslesen der Sensormesswerte
2. Modell zur Umrechnung in Strom unter Berücksichtigung eines Offsets
3. Darstellung des gemessenen Stroms und dessen zeitlicher Verlauf
4. Weitere sekundäre Anforderungen:
  - (a) Darstellung eines Heartbeats
  - (b) Nutzung des internen Watchdogs
  - (c) Nutzung von Button zum Anstoßen einer Kalibrierung

Im Folgenden wird ein Überblick über den Gesamtaufbau der Software gegeben sowie auf einzelne Implementierungsdetails genauer eingegangen. Beim Design der Softwarearchitektur wurde sich entschieden einzelne funktionale Blöcke wie den Heartbeat oder den Watchdog sowie hardware-spezifische Implementierungen in eigenen Bibliotheken zu kapseln. Die Abbildung 5 stellt die gekapselten Bibliotheken und ihre internen Abhängigkeiten untereinander dar. Dieser stark modulare Ansatz erlaubt die Wiederverwendung großer Programmteile auf anderer Hardware, nur eine Reimplementierung der hardwarenahen Funktionen ist nötig, oder in anderen Projekten auf gleicher Hardware.

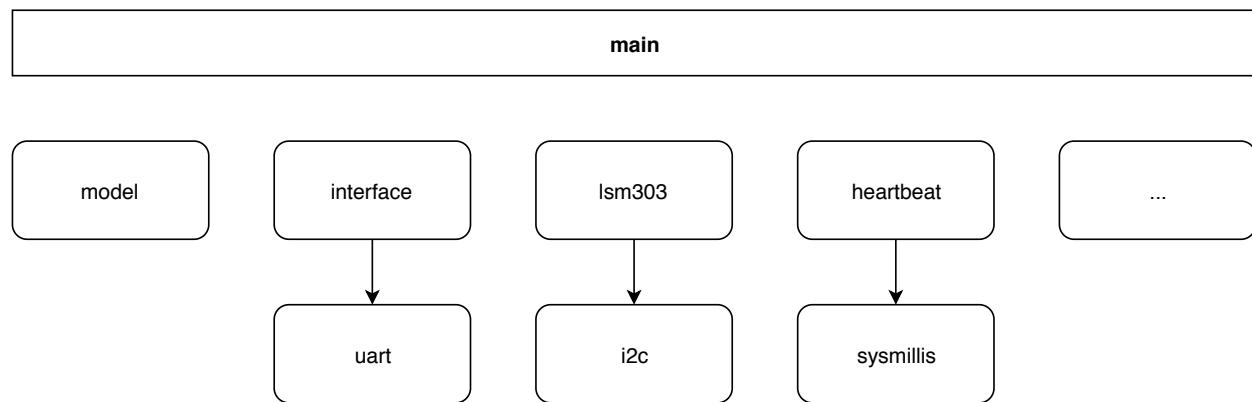


Abbildung 5: Software Übersicht

Die Bibliotheken folgen dabei zu einem großen Teil einer einheitlichen Struktur. Listing 1 ist der Header der *heartbeat* Bibliothek ohne Kommentare. Wenn eine Bibliothek eine `*_init()` Funktion

definiert, so muss sie vor der Benutzung anderer Bibliotheks Funktionen aufgerufen werden. Sie dient der internen Initialisierung der Bibliothek. Zusätzlich ist die Anwendung verantwortlich die **\*\_process()** Funktionen zyklisch aufzurufen. Dieses als Polling bekannte Konzept ermöglicht den Bibliotheken zu überprüfen, ob Arbeit vorliegt und diese zu bewältigen.

```
1 #pragma once
2
3 void heartbeat_init();
4 void heartbeat_process();
```

Listing 1: heartbeat.h ohne Kommentare

Um das Zusammenspiel der unterschiedlichen Bibliotheken und somit Abstrahierungsebenen zu veranschaulichen, ist in Abbildung 6 in einem Sequenzdiagramm die Initialisierung des Sensors dargestellt. Der zugehörige Sourcecode, welcher in der Main Funktion der Anwendung zu finden ist, wird in Listing 2 dargestellt. Zu Beginn initialisiert die Anwendung auf dem Mikrocontroller mit dem Aufruf **i2c\_init()** die hardwarenahe i2c Bibliothek. Diese kümmert sich um die Initialisierung der i2c Peripheral des Mikrocontrollers. Anschließend wird durch den Aufruf zur Initialisierung der *lsm303* Bibliothek (Sensor) unter Benutzung der i2c Bibliothek der Sensor aktiviert und dabei Einstellungen vorgenommen. Die *lsm303* Bibliothek bekommt hierfür einen Handler des Types **i2cDevice\_t** übergeben, welches zur Identifikation der i2c Schnittstelle dient. Soll die Sensor Bibliothek auf anderer Hardware verwendet werden, muss ausschließlich eine auf die neue Hardware angepasste i2c Bibliothek implementiert werden.

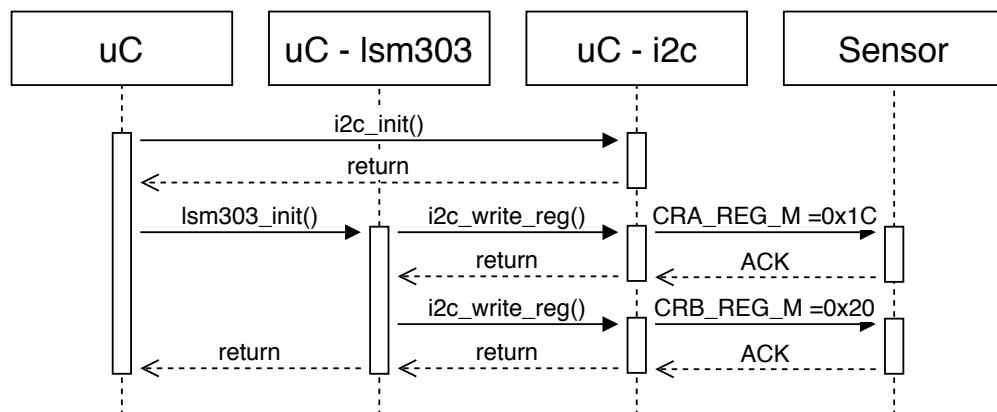


Abbildung 6: Initialisierung des Sensors

```
1 i2cDevice_t i2c2;
2 i2c_init(&i2c2);
3 lsm303dlhcSensor_t sensor;
4 lsm303dlhc_init(&sensor, &i2c2);
```

Listing 2: Initialisierung Code

Einen wesentlichen Teil für unsere Strommess-Anwendung nimmt die *model* Bibliothek ein. Sie rechnet die in *nG* vorliegenden Messwerte aus der *lsm303* Bibliothek in einen Strom um. Dafür nutzt sie neben einem festkodierten, da nur vom Abstand des Leiters bestimmten, Faktor einen

umgebungsabhängigen Offset. Um diesen, welcher der Stärke des Umgebungsfeldes entspricht (Nullpunktikalibrierung des Stroms), zu ermitteln, kann das Modell veranlasst werden eine Kalibrierung automatisch durchzuführen. Diese Funktionalität, welche durch die Main Methode bei einem Druck auf den SW1 Button der Evaluierungsplatine ausgelöst wird, versetzt die in dem Modell vorliegende Zustandsautomat in den Zustand *MODEL\_CALIBRATION\_ONGOING*. Wird nun das Modell mit neuen Magnetfelddaten versorgt, werden diese nicht zur Berechnung des Stroms sondern zur Ermittlung des Offsets verwendet. Dafür werden eine konfigurierbare Anzahl an Messdaten arithmetisch gemittelt. Nachdem genügend Messwerte zur Kalibrierung verwendet wurden, wechselt das Modell in den Zustand *MODEL\_CALIBRATION\_DONE*. Dieser Vorgang ist in Abbildung 7 dargestellt.

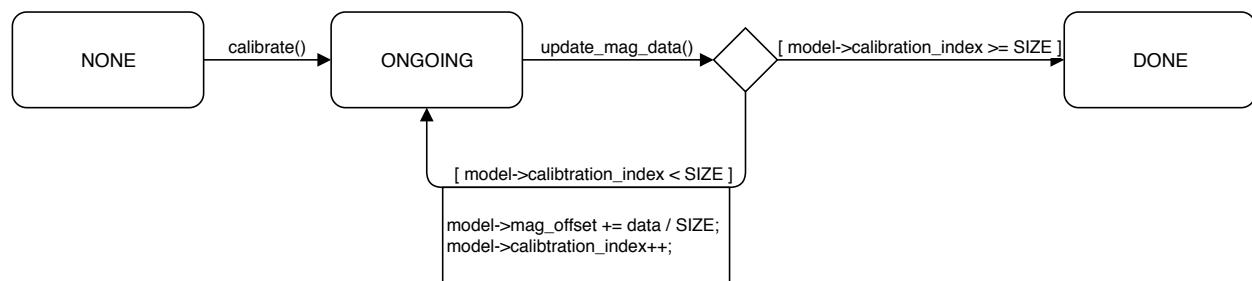


Abbildung 7: Zustandsautomat der Model Bibliothek

Die für das Modell notwendigen magnetischen Felddaten werden von dem Sensor abgefragt sobald dieser durch einen externen Interrupt mitteilt, dass neue Messdaten vorliegen. Dieses Vorgehen wird in einem Sequenzdiagramm in Abbildung 8 dargestellt. Diese abgefragten Messdaten werden nun an das Modell weitergereicht, welches diese entweder zur Kalibrierung verwendet oder den aktuellen Strom berechnet. Daraufhin werden vom Modell sowohl der Status des Zustandsautomaten als auch der aktuelle Stromwert abgefragt und mit einem Zeitstempel versehen an die interface Bibliothek weitergereicht. Diese packt die Daten in das in Listing 3 dargestellte Format und nutzt die *uart* Bibliothek um sie zu versenden.

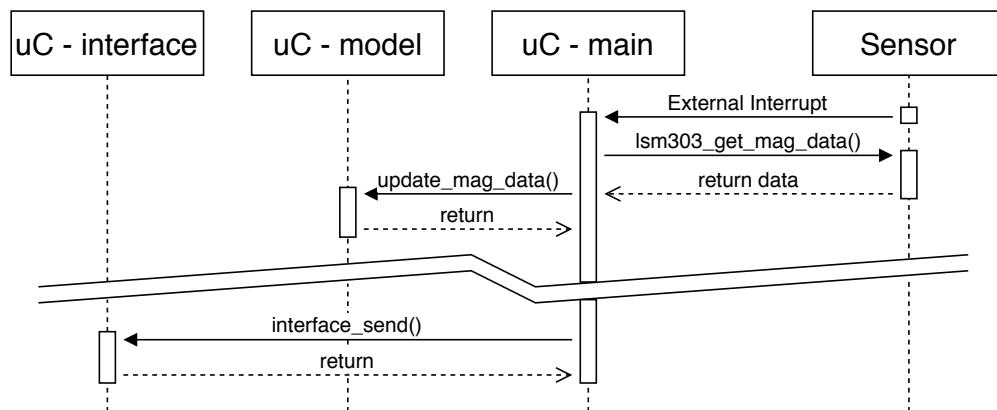


Abbildung 8: Ablauf Messdatenabfrage

```
1 /**
2 * @brief Structure used to pack the data
3 */
4 typedef union {
5     struct {
6         uint8_t magic_byte;
7         uint8_t status;
8         uint32_t timestamp;
9         float payload;
10    } __attribute__((packed));
11    uint8_t raw[10];
12 } InterfaceDataFrame_t;
```

Listing 3: Datenstruktur Binärinterface

Das Datenformat kodiert neben dem Payload, welcher dem Strom entspricht, und einem Zeitstempel auch den aktuellen Status des Modells sowie ein Magic Byte zur Erkennung des Startpunktes eines Paketes. Die Verwendung einer **union** erlaubt neben dem Zugriff auf einzelne Elemente der tieferliegenden **struct** auch einen byteweisen Rohzugriff, welcher unter anderem für das Versenden genutzt wird. Um ein automatisches Padding des Compilers, welches ungenutzte Speicherbereiche zwischen den Elementen der **struct** hinzufügen würde, zu unterbinden, muss die innenliegende **struct** als "packed" attribuiert werden.

Zur Darstellung der Stromwerte über Zeit wurde sich entschlossen ein Python Skript zu schreiben, welches auf einem angeschlossenen Computer die Werte über die serielle Schnittstelle ausliest und anzeigt. Die Darstellung dieser Werte ist in Abbildung 9 zu sehen.

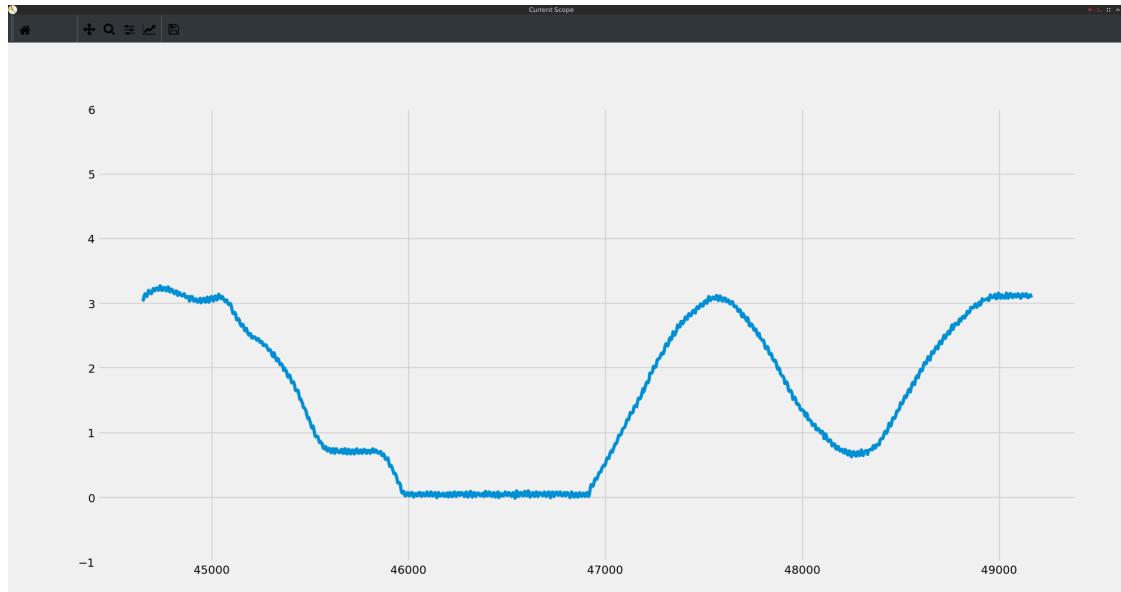


Abbildung 9: Screenshot Python Skript zur Darstellung

## 5 Validierung und Test

Um die Messdaten des Sensors zu validieren, wird mittels eines Labornetzteils ein definierter Strom durch den Leiter geschickt und dieser zusätzlich über einen Shunt mittels eines Rohde & Schwarz Scoperider 4001 Oszilloskopes gemessen. Ziel ist es, sowohl die Qualität der Messwerte des Sensors, als auch das Ansprechverhalten des Sensors zu überprüfen.

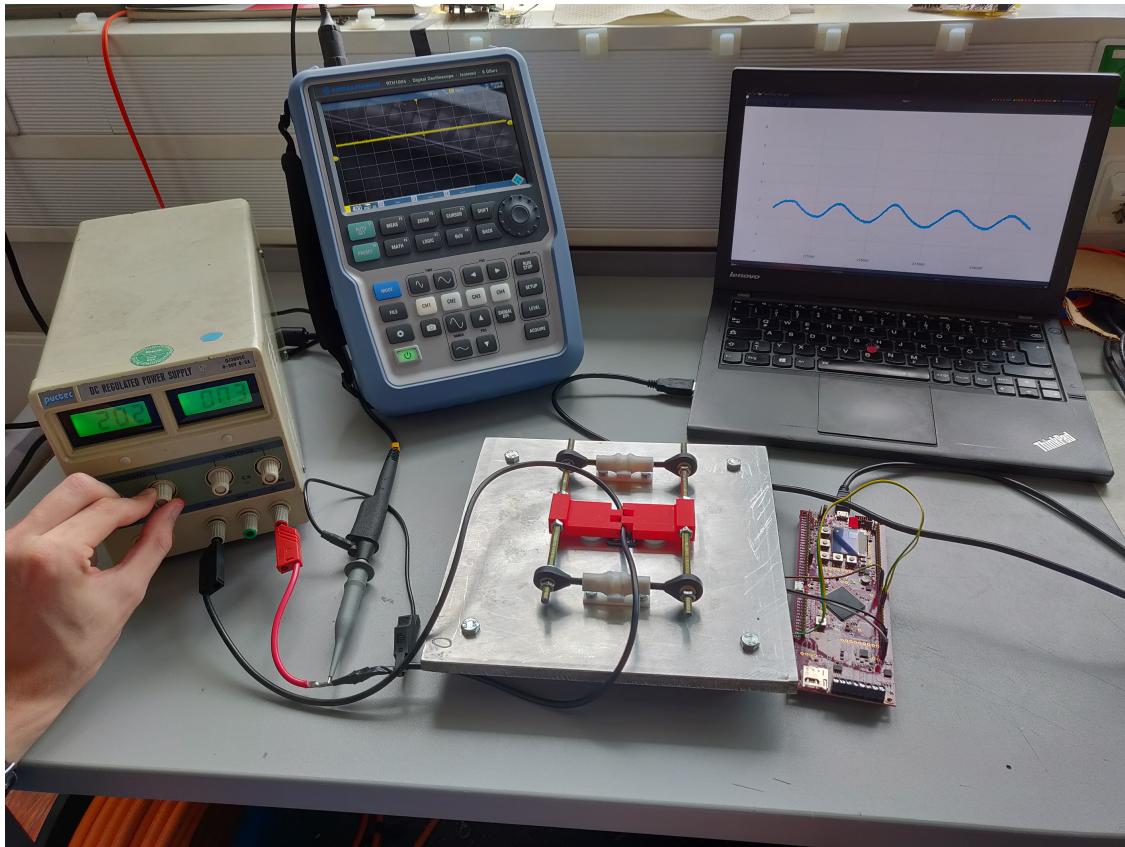


Abbildung 10: Versuchsaufbau

In der Abbildung ist in der Mitte die Halterung mit dem Sensor zu sehen, links wird mittels des Oszilloskops der Strom über einen Shunt gemessen, rechts werden die Sensordaten auf dem Laptop grafisch dargestellt.

Um das Ansprechverhalten des Sensors zu ermitteln, wurde sowohl das Ein- als auch das Ausschalten der Netzquelle mit beiden Messmethoden aufgezeichnet und die Daten in MatLab übereinandergelegt.

In Rot ist jeweils das Signal des Oszilloskops zu sehen, in Blau das Sensorsignal. Beim Einschalten zeigt sich, dass der Sensor dem Wert ebenso gut folgt wie das Oszilloskop. Jedoch macht sich die deutlich niedrigere Abtastrate bemerkbar.

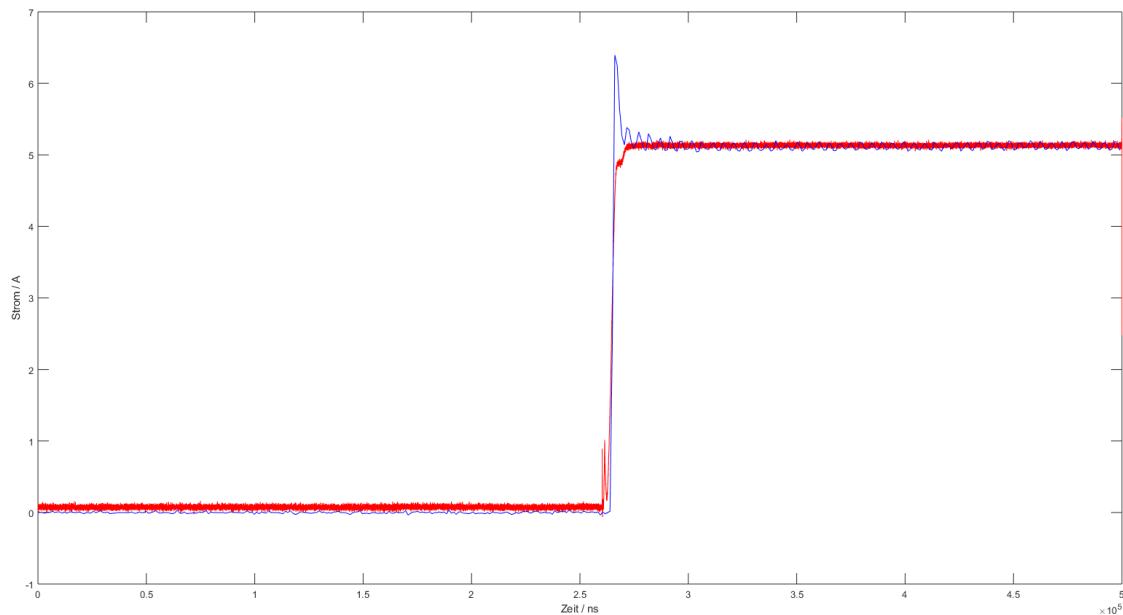


Abbildung 11: Einschaltvorgang

Gleiches gilt auch für den Abschaltvorgang. Das Oszilloskop löst sehr viel feiner auf, jedoch stimmen schon nach kurzer Zeit beide Werte wieder überein.

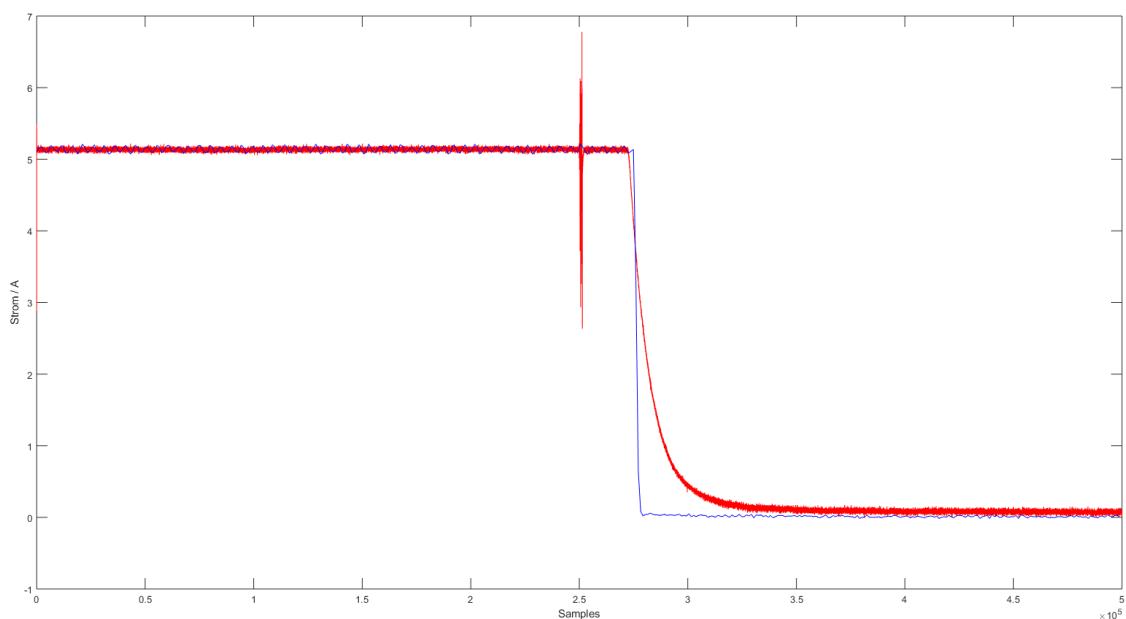


Abbildung 12: Abschaltvorgang

## Literaturverzeichnis

- [1] Stefan Valentinov Vutev Georgi Todorov Nikolov. Magnetic fields measurement with AMR sensors. *ANNUAL JOURNAL OF ELECTRONICS*, 2009.
- [2] Prof. Dr. Rudolf Gross. Vorlesungsskript Spinelektronik, 2005.
- [3] Junio Hamano. Git. <https://git-scm.com/>. Zuletzt aufgerufen 18.07.2019.
- [4] Uwe Loreit. Magnetoresistive Sensoren. <http://www.mr-sensor.de/>. Zuletzt aufgerufen 15.05.2019.
- [5] Evan Martin. Ninja build system. <https://ninja-build.org/>. Zuletzt aufgerufen 18.07.2019.
- [6] Jussi Pakkanen. Meson build system. <https://mesonbuild.com/>. Zuletzt aufgerufen 18.07.2019.
- [7] Dimitri van Heesch. Doxygen. <http://www.doxygen.nl/>. Zuletzt aufgerufen 18.07.2019.