

Entwicklung eines Schwingungsmesssystems für ein Radar

Sebastian Beyer

20. August 2012

Inhaltsverzeichnis

1 Abstrakt	3
2 Motivation und Hintergrund	4
3 Hintergrund: Beschleunigungssensoren	5
3.1 Piezoresistive Sensoren	6
3.2 Kapazitative Sensoren	6
4 Entwicklung des ersten Schwingungsmesssystems (BMA180)	9
4.1 Bosch BMA180	9
4.2 Arduino	9
4.3 Schnittstellen	10
4.3.1 I ² C	10
4.3.2 Serieller Port	10
4.4 Aufbau und Schaltung	11
4.4.1 Testaufbau	11
4.4.2 Fester Aufbau	12
4.5 Software	15
4.5.1 Arduino	15
4.5.2 PC Logger	18
5 Experimente in Waakirchen	18
6 Results	18

1 Abstrakt

2 Motivation und Hintergrund

3 Hintergrund: Beschleunigungssensoren

Beschleunigungssensoren messen grundsätzlich die Kraft, die auf ein System wirkt. Diese setzt sich aus der Gravitation und Trägheitskräften zusammen. Isoliert man die Trägheitskräfte, so lässt sich daraus die Translationsbeschleunigung bestimmen. Mithilfe dieser lässt sich die Positionsänderung des Systems berechnen. (Referenz?)

Es gibt verschiedene Prinzipien, mit denen die Beschleunigung bestimmt werden kann. Das verbreitetste ist das der Federwaage.

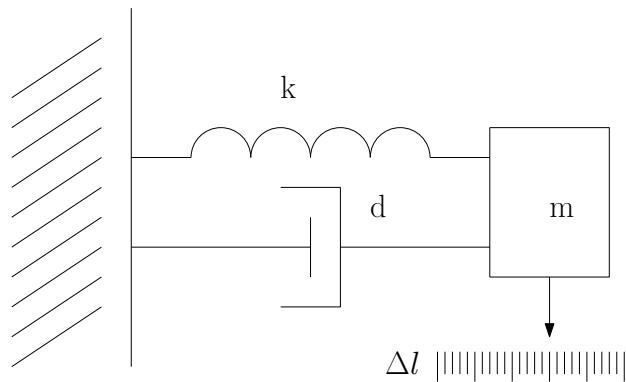


Abb. 1: Masse-Feder-System zur Beschleunigungsmessung. k , d , m und Δl bezeichnen Federkonstante, Dämpfung, Masse und Auslenkung. nach [Klingbeil, 2006].

Eine Masse m (auch seismische Masse genannt) ist über einer Feder mit der Federkonstante k mit einem festen Bezugspunkt verbunden. Die Auslenkung Δl ist proportional zur auf die Masse wirkenden Beschleunigung a .

$$a = \frac{k}{m} \cdot \Delta l \quad (1)$$

Zu beachten ist die Ausrichtung des Sensors, a entspricht immer der Projektion der Beschleunigung auf die Auslenkungsrichtung von m .

Durch geeignete Wahl von k , d und m lässt sich das System für den jeweiligen Fall so einstellen, dass die Ansprechzeit, Resonanz und Sensitivität optimal sind und es möglichst zu keinerlei Nachschwingen kommt.

Mittlerweile sind Beschleunigungssensoren in den meisten Fällen als MEMS realisiert (**Micro Electro Mechanical Systems**). Sehr kleine mechanische Elemente (1-100 Mikrometer) werden zusammen mit elektronischen Schaltungen auf einen Siliziumwafer aufgebracht. Dabei werden Techniken aus der Fabrikation von integrierten Schaltkreisen (ICs) verwendet, was dazu führt, dass komplizierte elektromechanische Systeme in winziger Größe und hoher Stückzahl hergestellt werden können. Der geringe Preis führt dazu, dass die Sensoren in mehr und mehr Anwendungen integriert werden (Autos, Smartphones, Quadrocopter...).

Die Messung der Auslenkung erfolgt im Wesentlichen durch zwei Verfahren:

3.1 Piezoresistive Sensoren

Piezoresistive Sensoren machen sich den Piezoelektrischen Effekt zunutze. In der Feder der Testmasse befinden sich Piezoelemente, welche sich bei Auslenkung verformen und damit ihren Widerstand ändern. Silizium ist ein geeignetes Material, da es sehr empfindlich und linear reagiert und gleichzeitig gut mit der MEMS Technik kombinierbar ist [Kanda, 1991].

3.2 Kapazitative Sensoren

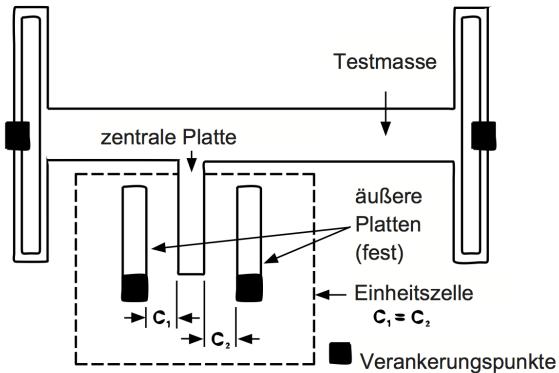


Abb. 2: Vereinfachtes Diagramm des ADXL05 in Ruhe [ADXL05, 1996]

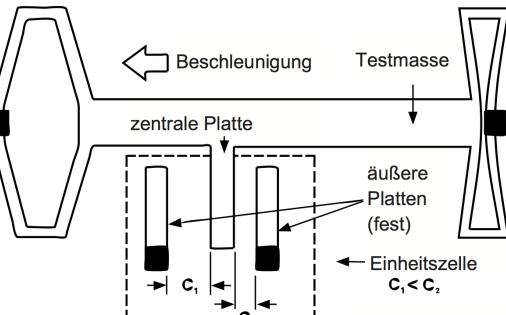


Abb. 3: Vereinfachtes Diagramm des ADXL05 während einer externen Beschleunigung [ADXL05, 1996]

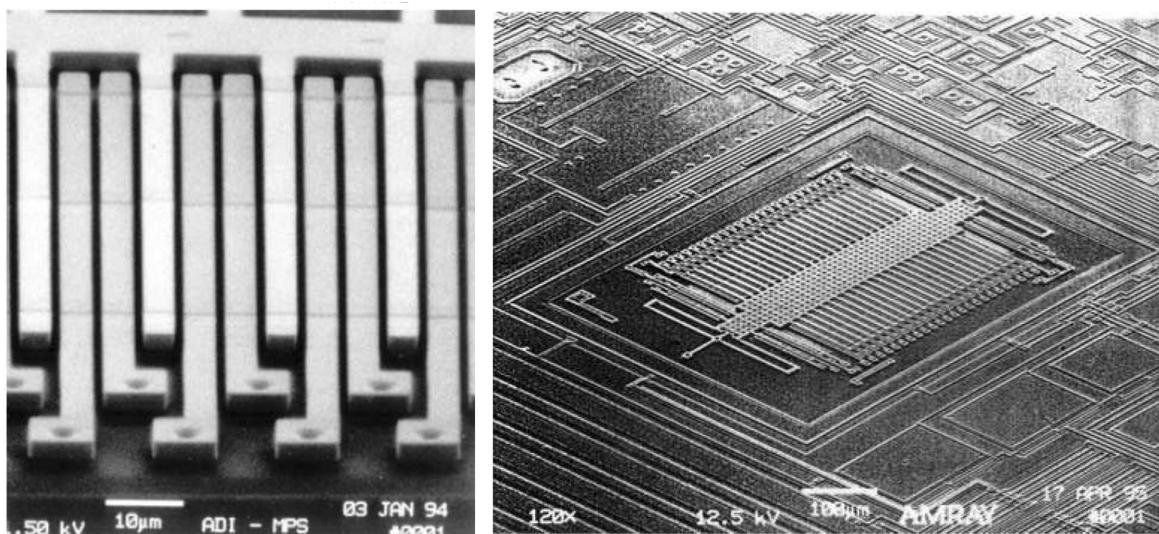


Abb. 4: ADXL05 unter dem Elektronenmikroskop, Beispiel für einen kapazitiven MEMS Beschleunigungssensor mit 46 Einzelzellen [Klingbeil, 2006].

Die Auslenkung lässt sich auch über eine Kapazitätsmessung bestimmen, wenn man je eine Elektrode an der Testmasse und eine am fixen Referenzpunkt anbringt.[Sherman

et al., 1992] Eine mögliche Realisierung ist in Abbildung 2 und Abbildung 3 zu sehen: Es handelt sich um einen sogenannten Differentialkondensatorsensor. Der Vorteil bei dieser Art von Sensoren ist die Möglichkeit zur Linearisierung und Kompensation [Schmidt, 2002]. (ERKLÄREN??)

Die seismische Masse befindet sich zwischen zwei fest verankerten Platten, die zusammen zwei Kondensatoren C_1 und C_2 bilden (Abbildung 2). Eine auftretende Beschleunigung führt also zu einer Auslenkung der Mittelelektrode des Kondensatorenpaars (seismische Masse) um die Länge $\pm\Delta l$ und damit zu einer symmetrischen Kapazitätsänderung um $\pm\Delta C$ (Abbildung 3). Dieser Aufbau ist eine sogenannte Einzelzelle. In einem Sensor befinden sich viele Einzelzellen, um die Kapazitätsvariation und damit die Sensibilität zu erhöhen.

Um diese Variation in ein elektrisches Ausgangssignal umzusetzen, wird eine sogenannte Brückenschaltung verwendet (Abbildung 5).

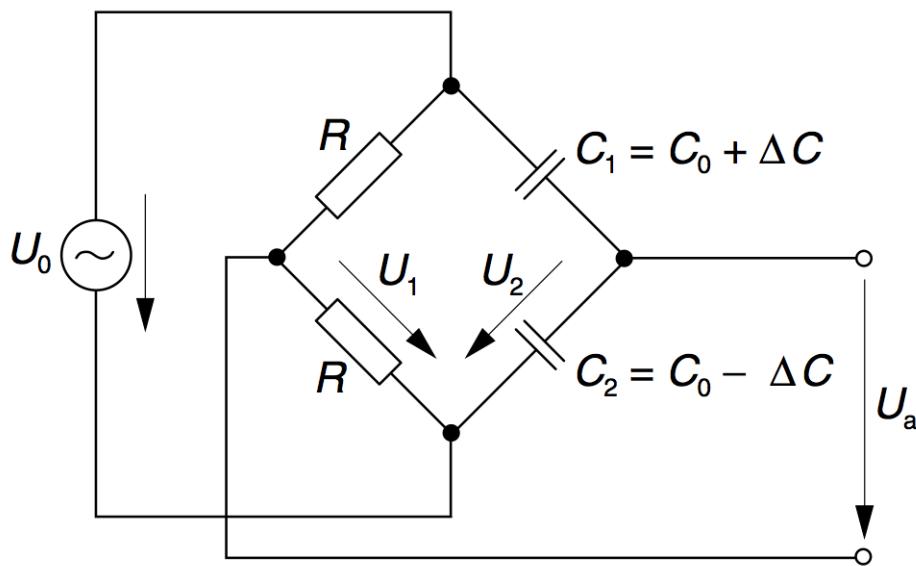


Abb. 5: Brückenschaltung mit Differentialkondensator (Die mit U_a verbundenen Elektroden von C_1 und C_2 bilden eine gemeinsame Platte) [Schmidt, 2002]

Wird nun eine Wechselspannung an die Brücke angelegt, so ergibt sich nach der Spannungsteilerregel für den Ausgang U_a :

$$U_a = U_1 - U_2 = U_0 \frac{R}{2R} - U_0 \frac{\frac{1}{j\omega C_2}}{\frac{1}{j\omega C_2} + \frac{1}{j\omega C_1}} \quad (2)$$

Durch Kürzen ergibt sich:

$$U_a = \frac{U_0}{2} - U_0 \frac{\frac{1}{C_2}}{\frac{1}{C_2} + \frac{1}{C_1}} = U_0 \left(\frac{1}{2} - \frac{C_1}{C_2 + C_1} \right) = \frac{U_0}{2} \left(\frac{C_2 - C_1}{C_2 + C_1} \right) \quad (3)$$

Mit $C_1 = C_0 + \Delta C = \varepsilon_0 \cdot \varepsilon_r \cdot A / (d_0 - \Delta l)$ und $C_2 = C_0 - \Delta C = \varepsilon_0 \cdot \varepsilon_r \cdot A / (d_0 + \Delta l)$ erhält man eine lineare Abhängigkeit der Ausgangsspannung U_a von der Auslenkung Δl :

$$U_a = -U_0 \frac{\Delta l}{2d_0} \quad (4)$$

Aus [Schmidt, 2002].

Diese Spannung lässt sich nun digitalisieren und auslesen. In vielen MEMS Bausteinen ist bereits ein integrierter Analog Digital Wandler eingebaut, sodass die Messwerte direkt digital abrufbar sind.

4 Entwicklung des ersten Schwingungsmesssystems (BMA180)

Um abschätzen zu können wie groß die auftretenden Beschleunigungen an der Radaranenne sind, habe ich damit begonnen einen günstigen, schnell zu verwendenden Prototypen zu entwickeln. Dies ist vor Allem wichtig, um entscheiden zu können, für welchen Messbereich und welche Frequenzen der eigentliche Sensor ausgelegt sein muss.

4.1 Bosch BMA180

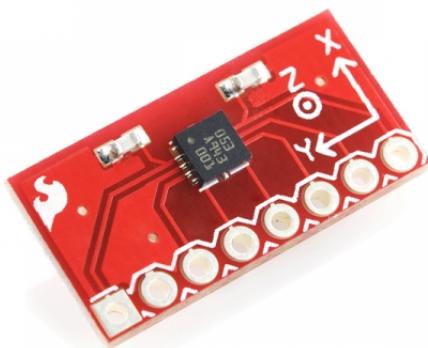


Abb. 6: BMA180 Breakoutboard von Sparkfun

Die Wahl fiel auf einen *Bosch BMA180* digitalen, dreiachsigen MEMS Beschleunigungssensor. Er verfügt über einen eingebauten 14-bit Analog-Digital Wandler und sieben per Software verstellbaren Messbereiche von ± 1 bis $\pm 16g$. Die Kommunikation kann über SPI oder I2C erfolgen. Analogfilter

Ich nutze ein Breakoutboard von Sparkfun, da so das knifflige SMD-Löten entfällt und bereits zwei spannungsstabilisierende Kondensatoren eingebaut sind.

4.2 Arduino

Arduino ist eine auf ATmega Mikroprozessoren basierende Open-Source Entwicklungsplattform zur Verarbeitung von analogen und digitalen Signalen. Die Programmierung kann über eine eigene Entwicklungsumgebung in einer an *Processing*¹ angelehnten Sprache erfolgen, die im Prinzip ein vereinfachtes C/C++ darstellt.

Die Plattform ist auf Prototyping und Experimente ausgelegt. Es ist bereits ein Bootloader vorinstalliert, so kann die Programmierung direkt über die serielle Schnittstelle erfolgen. Die Boards machen die meisten Pins des Atmegas für eigene Schaltungen verfügbar, in den gängigen Boards sind das 14 Pins, die frei als Ein- oder Ausgänge genutzt werden können. Die Stromversorgung kann über USB oder eine externe 5V Quelle erfolgen. Als weitere Interfaces werden SPI, ICSP und I2C angeboten.

Der ATmega arbeitet mit 16 MHz und hat einen geringen Energieverbrauch.

¹LINK?

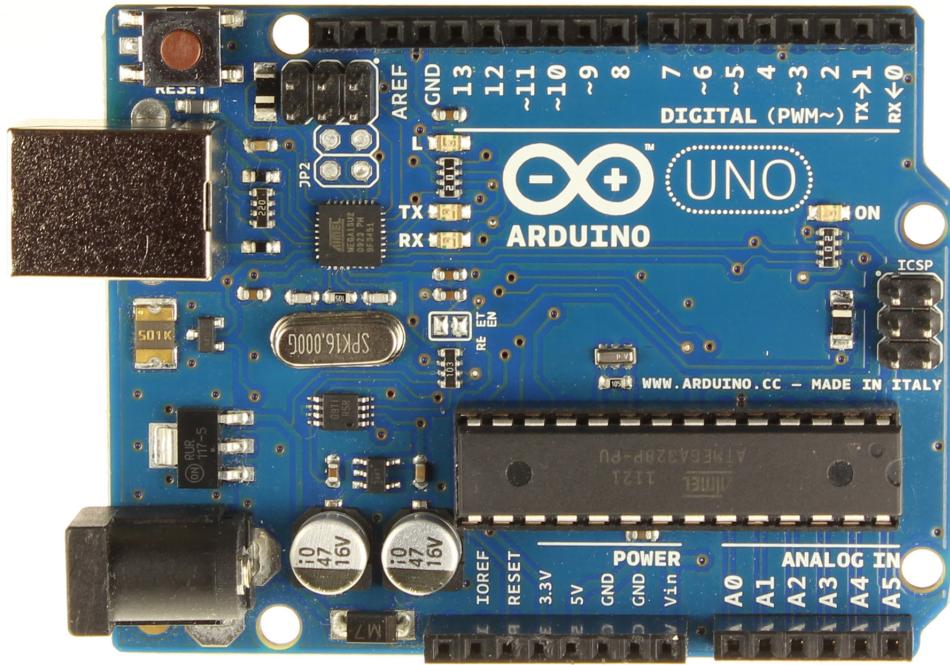


Abb. 7: Arduino UNO

4.3 Schnittstellen

4.3.1 I²C

Die Kommunikation zwischen Beschleunigungssensor und Arduino erfolgt über den I²C Bus[NXP, 2012]. Dabei handelt es sich um einen von Phillips entwickelten seriellen Datenbus, der ursprünglich entwickelt wurde, um Chips in Fernsehgeräten steuern zu können. Inzwischen ist das Patent ausgelaufen und er wird in vielen Hardwareprojekten verwendet, da er sehr leicht zu verstehen und zu verwenden ist.

Ich benutze die Arduino Wire Library [Arduino, 2012], welche die gesamte Kommunikation über i²C steuert und einfache Funktionsaufrufe zur Verfügung stellt.

4.3.2 Serieller Port

Zum einfachen Anschluss des Arduinos an einen PC oder Datenlogger wird eine serielle Schnittstelle nach RS232 (REFERENZ) verwendet. Auf dem Arduino befindet sich ein ATmega16U2, welcher die seriellen Signale in USB umwandelt und dafür sorgt, dass der Arduino am PC als virtueller COM-Port erscheint.

4.4 Aufbau und Schaltung

4.4.1 Testaufbau

Um die korrekte Verschaltung zu überprüfen und die Software für das Auslesen der Daten zu entwickeln, habe ich zunächst auf dem Breadboard gearbeitet. Der dazu verwendete Schaltplan ist in Abbildung 10 zu sehen.

Da der Sensor im I²C Modus betrieben werden soll, ist CS auf High (3.3V) geschaltet. Der SDO-Pin setzt damit dann die Adresse des Chips, wie aus Tabelle 1 zu entnehmen ist. Ist dieser Low (GND), so ist die Adresse 0x40.

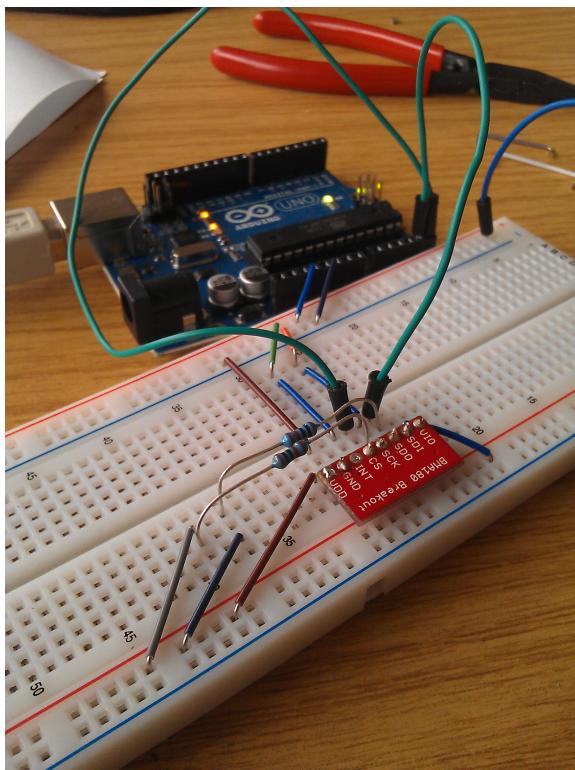


Abb. 8: Testaufbau auf dem Breadboard



Abb. 9: Testplatine am Radar

SPI mode	I2C mode
SDI input	SDA birectional (!)
SDO output	ADDR adress bit, input
SCLK input	SCL input
CSB chip select, input	I2C mode select, input

Tab. 1: BMA180 Pinbelegung für SPI und I²C Modes [Sensortec, 2009]

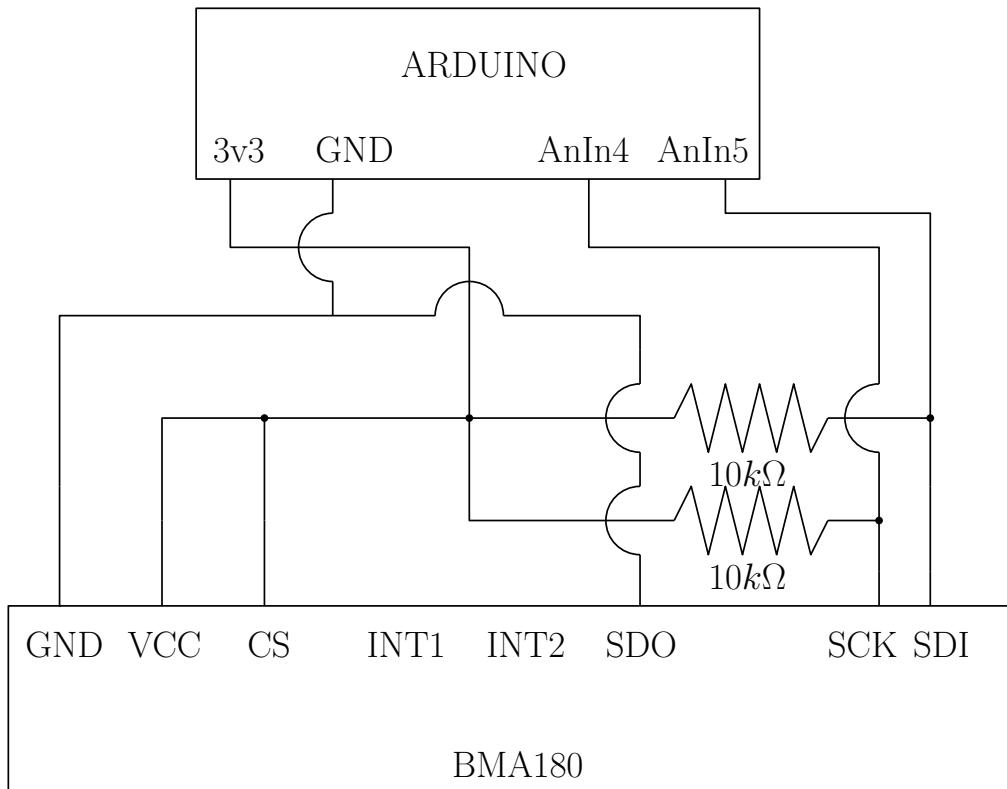


Abb. 10: Schaltplan Testaufbau. Die Interrupts sind nicht verbunden, VCC und CS sind auf 3.3V geschaltet, GND und SDO auf die Masse des Arduinos gezogen, die I²C Datenleitungen SCK und SDI sind mit den Arduinopins A4 und A5 verbunden, wobei zusätzlich 10kΩ Pull-Up-Widerstände eingebaut sind.

4.4.2 Fester Aufbau

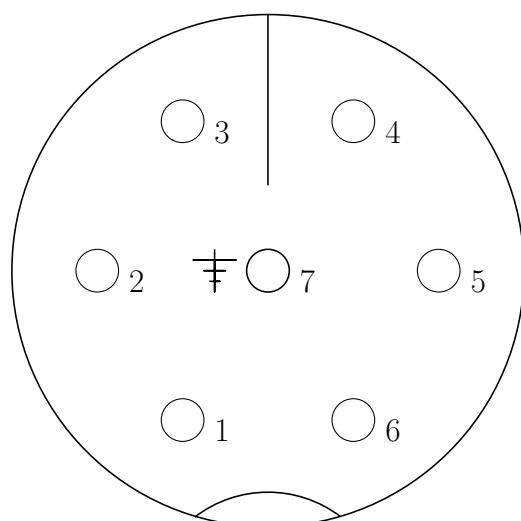
Um das System wirklich praktisch nutzen zu können muss er natürlich fest aufgebaut werden und mit einem Gehäuse versehen werden, das es erlaubt, ihn fest an einem Testobjekt anzubringen und ihn gleichzeitig vor Schäden durch mechanische oder witterungsbedingte Einflüsse schützt.

Um die eigentliche Sensoreinheit möglichst kompakt zu halten, habe ich mich entschieden, den Beschleunigungssensor vom Arduino zu trennen und auf eine kleine Lochrasterplatine zu löten. Diese wird in einen festen Block aus Polyurethanharz [Components, 2010] eingegossen, womit sie gleichzeitig gut geschützt und leicht anzubringen ist. (Abbildung 11)

Der Arduino ist in ein ABS Gehäuse [Enclosures, 2005] eingebaut (Abbildung 14 und 15). Die I²C Verbindung mit dem Sensor erfolgt über einen Hirschmannstecker™ mit 7 Polen, von denen 4 beschaltet sind, wie in Abbildung 13 zu sehen ist. Mit dem Logger ist der Arduino über ein USB-Kabel verbunden, über das die Kommunikation per RS232 geführt wird.



Abb. 11: BMA180 in Polyurethanharz eingegossen, die Kunststoffform wurde von der technischen Werkstatt des Geomatikums gefertigt.



Pin	I2C	Arduino	BMA180	Farbe
1	3.3V	3.3V	VDD	Rot
2				
3	SDA	A4	SDI	Gelb
4	SCL	A5	SCK	Grün
5				
6				
7	GND	GND	GND	Schwarz

Abb. 13: I²C Beschaltung des HirschmannsteckersTM

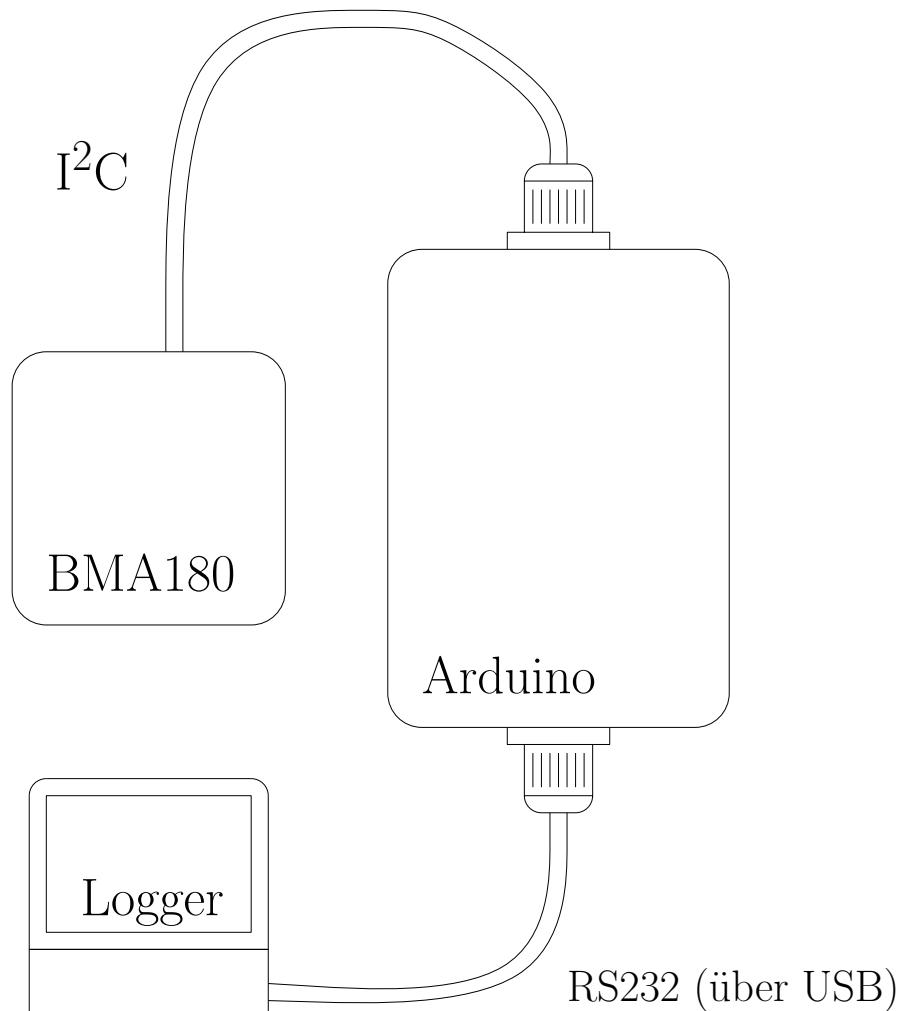


Abb. 12: Verbindungen zwischen BMA180, Arduino und Logger per I^2C bzw. RS232

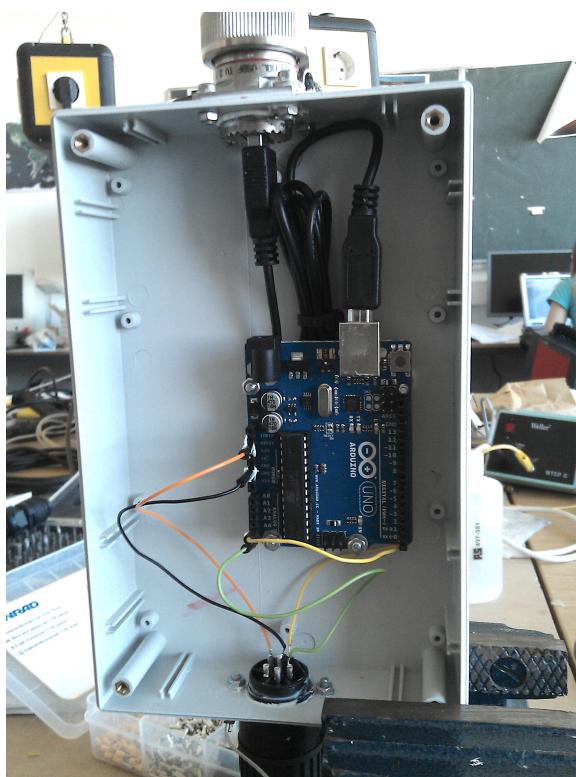


Abb. 14: ABS Box, welche den Arduino enthält



Abb. 15: ABS Box geschlossen im Witterungstest

4.5 Software

Die Software besteht aus zwei Teilen. Zunächst der Code auf dem Arduino, welcher die Daten vom Sensor abfragt und konvertiert und dann der Teil auf dem Logger, der die Messungen über den Seriellen Port erhält und diese gemeinsam mit der jeweils aktuellen Uhrzeit in eine Datei schreibt.

Bei der Erläuterung des Codes möchte ich mich auf die wesentlichen Punkte beschränken. Die gesamten Quelltexte befinden sich im Anhang.

4.5.1 Arduino

Der Arduino ist für die Kommunikation mit dem BMA180 über I²C zuständig. Für diese Aufgabe existiert eine Arduino Bibliothek, welche ich angepasst und verwendet habe (REFERENZ!!) Ich möchte hier die einzelnen Schritte beschreiben, die notwendig sind, um eine Messung zu erhalten und diese an den Logger weiterzuleiten.

Die Steuerung des BMA180 erfolgt über direkte Zugriffe auf die Register. Entweder man schreibt einen neuen Wert in ein Register oder man liest ein Register aus. Es gibt fünf Arten von Registern - Test, control, image, status und data. Davon interessieren uns nur die **control** und die **data** Register. Weitere Informationen über die zusätzlichen Register lassen sich im Datenblatt [Sensortec, 2009] nachschlagen.

Die control Register dienen dazu, den Beschleunigungssensor zu steuern, vornehmlich also den Messbereich festzulegen und eventuell einen zusätzlichen digitalen Bandfilter einzustellen. Dazu müssen Werte in Register hineingeschrieben werden. Dazu geht man folgendermaßen vor:

In Register schreiben

1. Übertragung initiieren (mit der Adresse des Sensors)
2. Adresse des Registers senden, in welches wir schreiben möchten
3. Daten senden, die wir in das Register schreiben möchten
4. Übertragung beenden

Da ein Register immer ein Byte (8 Bit) groß ist, finden sich oftmals verschiedene Einstellungen in einem Register um Platz zu sparen. Einen korrekten Schreibvorgang auszuführen wird dadurch komplexer. Daher möchte ich den Prozess am Beispiel der Messbereichseinstellung demonstrieren.

offset_y	30h	offset_x<11:4> (msb)	readout_12bit	80h
offset_x	38h	offset_x<6:0> (msb)		00h
offset_t	37h			00h
offset_ls2	36h	offset_z<3:0> (lsb)		00h
offset_ls1	35h	offset_x<3:0> (lsb)	range<2:0>	00h
gain_z	34h	gain_z<6:0>	smp_skip	80h
gain_y	33h	gain_y<6:0>	wake-up	80h
gain_x	32h	gain_x<6:0>	shadow_dis	80h
gain_t	31h	gain_t<4:0>	dis_reg	80h
			tapsens_dlr<2:0>	80h

Abb. 16: Memory Map BMA180, Ausschnitt [Sensortec, 2009]

Die relevanten Bits sind also im Register `0x35`², Bit eins, zwei und drei³. Hier die entsprechenden Zeilen aus dem Quellcode:

```
1 void BMA180:: setGSensitivity(GSENSITIVITY maxg) //1, 1.5 2 3 4 8 16
2 {
3     setRegValue(0x35, maxg<<1,0xF1);
4     gSense = maxg;
5 }
```

Listing 1: Funktion zum Einstellen des Messbereichs aus der BMA180 Bibliothek(REF!!)
maxg ist eine dem Wertebereich zugeordnete Binärzahl zwischen 000 und 110

Die aufgerufene Funktion `setRegValue` bekommt als Argumente das zu schreibende Register (`0x35`), den zu schreibenden Wert (`maxg << 1`) und zusätzlich eine Maske, die angibt, welche Bits des Registers nicht verändert werden sollen mit.

Da das letzte Bit (Bit Null) eine andere Funktion hat, 'schieben' wir `maxg` mit Hilfe der *bitshift Operation* `<<` um ein Bit nach links.

Bei `gSense` handelt es sich um eine Statusvariable, die später benutzt wird, um den genutzten Wertebereich mit zu dokumentieren.

```
1 void BMA180:: setRegValue(int regAdr, int val, int maskPreserve)
2 {
3     int preserve=getRegValue(regAdr);
4     int orgval=preserve & maskPreserve;
5     Wire.beginTransmission(address);
6     Wire.write(regAdr);
7     Wire.write(orgval|val);
8     int result = Wire.endTransmission();
9     checkResult(result);
10 }
```

Listing 2: Funktion zum Schreiben eines Registers aus der BMA180 Bibliothek(REF!!)

Nun lesen wir zunächst den bisherigen Wert des Registers ein (Zeile 3) und maskieren ihn mit der übergebenen Maske, so dass wir alle Bits übernehmen, außer die, welche wir verändern wollen (Zeile 4). Dann schreiben wir in das Register, wobei wir nach dem oben beschriebenen Schema vorgehen. Beim Senden der Daten verknüpfen wir jedoch `orgval` mittels eines bitweisen *Ors* mit dem zu schreibenden Wert (Zeile 7). So haben wir sicher gestellt, dass wir nur die relevanten Bits verändern.

In Abbildung 17 ist das Vorgehen noch einmal dargestellt.

²Ein vorangestelltes `0x` bedeutet, dass es sich um eine Zahl im Hexadezimalsystem handelt

³Es wird von rechts nach links gezählt und mit 0 begonnen

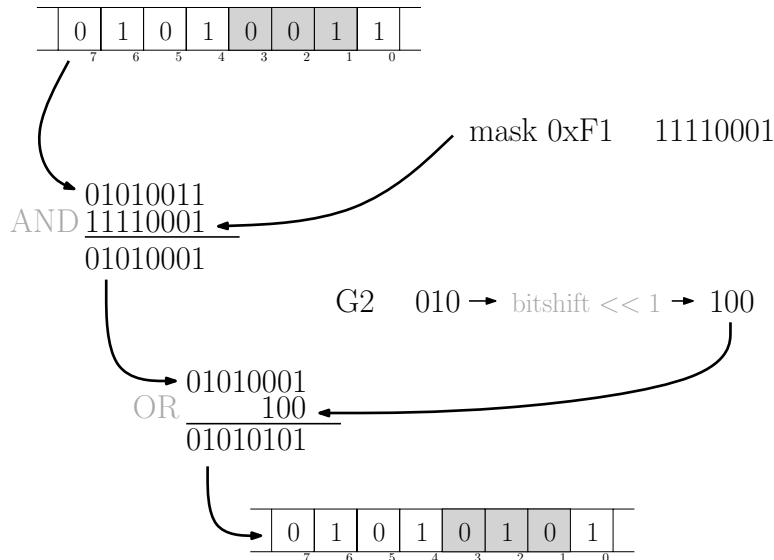
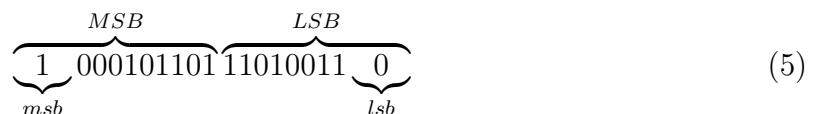


Abb. 17: Beispiel eines Schreibvorgangs in das Register 0x35. Nur die grau markierten Bits werden verändert.

Daten aus Registern lesen

In die data Register werden die gemessenen Beschleunigungsdaten geschrieben. Da der A/D-Wandler des BMA180 eine Auflösung von 14 Bit hat, existieren pro Achse 2 Register mit jeweils 8 Bit. Hier benötigen wir das Konzept der **Bitwertigkeit**:

In einer Zahl haben die Stellen unterschiedlichen Wert. Verändert man zum Beispiel bei der Zahl 4653 die letzte Stelle, so ist der Unterschied maximal 9. Wird hingegen die dritte Stelle (von hinten) verändert, so kann der Unterschied bis zu 900 betragen. Analog verhält es sich mit den binären Zahlen. Dabei bezeichnen wir das höchstwertige Bit als **most-significant-bit (msb)** und das geringwertige Bit als **least-significant-bit (lsb)**. Analog dazu spricht man vom höchstwertigen Byte **most-significant-byte (MSB)** und geringwertigen Byte **least-significant-byte (LSB)**.



Die gemessene Beschleunigung ist also aufgeteilt und muss im Programm wieder zusammengesetzt werden.

Um die Beschleunigungsdaten vom Sensor zu laden, ist so vorzugehen:

1. Übertragung initiieren (mit der Adresse des Sensors)
2. Adresse des Registers senden, bei dem wir **anfangen** wollen zu lesen
3. Übertragung beenden
4. So viele Bytes abrufen, wie wir haben möchten

5. Übertragung beenden

```

1 void BMA180::readAccel()
2 {
3     unsigned int result;
4
5     Wire.beginTransmission(address);
6     Wire.write(0x02);
7     Wire.endTransmission();
8     Wire.requestFrom((int)address, 7);
9     if(Wire.available() == 7)
10    {
11        int lsb = Wire.read() >> 2;
12        int msb = Wire.read();
13        x = (msb << 6) + lsb;
14        if (x & 0x2000) x |= 0xc000; // set full 2 complement for neg values
15        lsb = Wire.read() >> 2;
16        msb = Wire.read();
17        y = (msb << 6) + lsb;
18        if (y & 0x2000) y |= 0xc000;
19        lsb = Wire.read() >> 2;
20        msb = Wire.read();
21        z = (msb << 6) + lsb;
22        if (z & 0x2000) z |= 0xc000;
23        temp = Wire.read();
24        if (temp & 0x80) temp |= 0xff00;
25    }
26    result = Wire.endTransmission();
27 }
```

Listing 3: Funktion zum Auslesen der Beschleunigungsdaten

Das Register, bei dem wir beginnen wollen zu lesen, ist `0x02`, (Zeile 6) und wir benötigen die nächsten 7 Bytes (Zeile 8). Jeweils zwei Bytes MSB und LSB für die drei Achsen und als letztes Byte lesen wir zusätzlich die aktuelle Temperatur aus. Diese wird jedoch weiterhin nicht mehr verwendet.

4.5.2 PC Logger

5 Experimente in Waakirchen

6 Results

Eruption ist zu sehen! Schüssel bewegt sich um x m/s durch Eruption! Im auflösbaren Bereich? Arbeitsweise des Radars -> Tina fragen
 Korrelation zwischen Geschwindigkeit (Radargemessen) und gemessener Beschleunigung?

Literatur

Datasheet ADXL05. *1g to 5g Single Chip Accelerometer with Signal Conditioning*, 1996.

Arduino. Arduino wire library, 2012. URL
<http://www.arduino.cc/en/Reference/Wire>.

RS Components. *Datasheet Polyurethane*, 2010.

BOSS Enclosures. *ABS Gehäuse*, 2005.

Yozo Kanda. Piezoresistance effect of silicon. *Sensors and Actuators A: Physical*, 28(2): 83–91, July 1991.

Lasse Klingbeil. *Entwicklung eines modularen und skalierbaren Sensorsystems zur Erfassung von Position und Orientierung bewegter Objekte*. PhD thesis, Universität Bonn, 2006.

NXP. *UM10204 I2C-bus specification and user manual*. NXP, rev4 edition, February 2012.

Schmidt. *Sensorschaltungstechnik*. Vogel, 2002.

Bosch Sensortec. *BMA180 Digital, triaxial acceleration sensor Data sheet*, 2.1 edition, December 2009.

S.J. Sherman, W.K. Tsang, T.A. Core, R.S. Payne, D.E. Quinn, K.H.-L. Chau, J.A. Farash, and S.K. Baum. A low cost monolithic accelerometer; product/technology update. In *Electron Devices Meeting, 1992. IEDM '92. Technical Digest., International*, pages 501–504, 1992.