Neologismus

Ausgabe 04/2014

Bild: Michael Thies

„Der GAU für Webverschlüsselung“ – S. 12

Foto: Lukas Heimann

Quadrokopter, Teil II – S. 6

Foto: Marcel Hörz (Bearbeitung: Michael Thies)

Langzeittestbericht zur smarten Uhr – S. 11

Inhaltsverzeichnis

Natur- und Formalwissenschaft

Sei Epsilon kleiner Null ...

Teil 5: Gruppen und Körper

von Florian Kranhold

Ist Ihnen in den letzten Artikeln etwas aufgefallen? Wir haben bisher gar nicht gerechnet. Wir haben uns überlegt, wann Aussagen wahr und falsch sein können, wir haben Mengen bearbeitet und betrachtet, haben darauf zwischen Elementen Relationen hergestellt und schließlich zwischen zwei Mengen eine Abbildung gebastelt, die die Mengen auf eine bestimmte Weise vergleicht.

An keiner dieser Stellen haben wir uns Gedanken darüber gemacht, wie man irgendetwas addiert oder multipliziert; wir haben keine Wurzeln gezogen oder Logarithmen berechnet, nichts dergleichen. Dafür aber ist unsere bisherige Theorie viel allgemeiner, denn in der Schulmathematik untersucht man nur „langweilige“ Objekte, nämlich irgendwelche Zahlen. Bisher können wir Äpfel auf Birnen abbilden, oder Mengen von Funktionen schneiden; wir könnten Äquivalenzrelationen auf der Menge aller Abbildungen von A nach B definieren und viele andere solcher lustigen Konstrukte.

Nun wollen wir aber rechnen, dabei allerdings nach wie vor so allgemein wie möglich bleiben. Wir überlegen uns also: Mit welchen Objekten kann man auf welche Weise auch immer gescheit rechnen? Und wie können wir sicherstellen, dass da ein Objekt herauskommt, mit dem wir auch etwas anfangen können? Offensichtlich können wir nicht Äpfel mit Birnen mutliplizieren und hoffen, dass da ein Pferd herauskommt, zumindest nicht, wenn wir das nicht so sonderbar definieren.

Eine Rechnung im einfachsten Sinne ist eine Operation, bei der ich, einfach gesprochen, „zwei Objekte 'reintue“ und „ein Objekt 'rauskriege“. Am besten sollen die alle vom selben Typ sein. Wenn ich also eine Menge M von Objekten habe und mit diesen rechnen will, dann ist eine sinnvolle Rechnung eine Abbildung $*$, die jedem Paar $(m_1, m_2) \in M \times M$ ein Ergebnis $m_1 * m_2 \in M$ zuordnet.

Nun können wir verschiedene Mengen mit einer solchen Abbildung, die wir *Verknüpfung* nennen, betrachten. Dann können wir zum Beispiel schauen, ob wir beim Rechnen umklammern können, ob also gilt $m_1 * (m_2 * m_3) = (m_1 * m_2) * m_3$. Auch interessant ist die Frage, ob es ein Objekt gibt, dass beim Verknüpfen alle anderen unverändert lässt, also ein $e \in M$, sodass $m * e = e * m = m$ für alle $m \in M$, so etwas wie die Null beim Addieren oder die Eins beim Multiplizieren. Wir nennen dieses *neutrales Element*.

Schließlich können wir uns noch fragen, ob es zu jedem $m \in M$ ein Element $m' \in M$ gibt, sodass wir beim Verknüpfen dieser bei dem neutralen Element landen, so etwas wie die Gegenzahl beim Addieren oder der Kehrbruch beim Multiplizieren. Wir nennen diese *inverse Elemente*.

Wenn wir eine Menge haben wollen, bei der all dies erfüllt ist, dann muss man sich schon genau überlegen, was da so drin ist. Die Menge $\{1, 3, 4\}$ zusammen mit + ist offenbar ganz schlecht: Wenn ich $1 + 4$ rechne, erhalte ich 5 und falle bereits aus der Menge heraus. Die Menge $\mathbb{N} = \{1, 2, ...\}$ ist zum Addieren ganz gut, hat aber kein neutrales Element, weil die 0 fehlt. Die Menge $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ mit der Addition hat zusätzlich ein neutrales Element, aber die Inversen, also hier die Gegenzahlen, fehlen.

Auf $\mathbb{Z} = \{..., -1, 0, 1, ...\}$ schließlich haben wir alles, was wir wollen. Versuchen wir aber, hierauf zu mutliplizieren, gehen uns irgendwann die Inversen aus: Das multiplikativ neutrale Element ist offensichtlich die 1, denn $1 \cdot a = a \cdot 1 = a$. Für viele ganze Zahlen, nämlich für alle außer 1 und -1, finden wir aber keine andere ganze Zahl, sodass beide miteinander multipliziert 1 ergeben. Wir bräuchten hierfür die Brüche und für die 0 werden wir nie ein solches Element finden.

Nennen wir $\mathbb{Q}$ die Menge aller Brüche. Hierauf können wir munter Addieren und alles machen, was wir wollen, wie oben beschrieben. Wir können auch multiplizieren und die Menge $\mathbb{Q} \setminus \{0\}$ zusammen mit der Multiplikation erfüllt wieder alles, was wir wollen. Das scheint also bereits eine ganz besondere Menge zu sein, weil wir hier fast uneingeschränkt *zwei* Operationen drauf ausführen können.

Wir wollen diese Strukturen ein bisschen besser verstehen.

5 Gruppen und Körper

Definition 5.1. Sei A eine Menge. Eine Abbildung

$$\begin{aligned} \kappa : A \times A &\to A \\ (a_1, a_2) &\mapsto \kappa(a_1, a_2) =: a_1 * a_2 \end{aligned}$$

heißt *Verknüpfung* auf A.

Definition 5.2. Eine *Gruppe* $(G, *)$ ist eine nichtleere Menge G zusammen mit einer Verknüpfung

$$* : G \times G \to G, (g_1, g_2) \mapsto g_1 * g_2$$

und folgenden Eigenschaften:

(G1) „$*$“ ist *assoziativ*, d. h. für alle Elemente $g_1, g_2, g_3 \in G$ gilt: $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$

(G2) G besitzt ein *neutrales* Element e_G, sodass $e_G * g = g = g * e_G$.

(G3) Jedes $g \in G$ besitzt ein *inverses* Element g', sodass gilt $g * g' = e_G = g' * g$

Eine Gruppe heißt *abelsch*, wenn gilt:

(Ab) „$*$“ ist *kommutativ*, d. h. für alle $g_1, g_2 \in G$ gilt $g_1 * g_2 = g_2 * g_1$

Bemerkung 5.3. Je nach dem, was für Gruppen untersucht werden, haben sich verschiedene Schreibweisen etabliert:

(i) *Allgemeine Schreibweise*
Die Verknüpfung heißt $*$, das neutrale Element e_G und das zu $g \in G$ inverse Element g^{-1}.

(ii) *Additive Schreibweise*
Die Verknüpfung heißt $+$, das neutrale Element 0_G (*Nullelement*) und das zu $g \in G$ inverse Element $-g$.

(iii) *Multiplikative Schreibweise*
Die Verknüpfung heißt $\cdot$ (oder wird nicht notiert), das neutrale Element 1_G (*Einselement*) und das zu $g \in G$ inverse Element g^{-1}.

Beispiel 5.4.

(i) $(\mathbb{N}, +)$ ist keine Gruppe, da das neutrale Element 0 nicht in $\mathbb{N}$ enthalten ist. $(\mathbb{N}_0, +)$ ist keine Gruppe da es nicht zu jedem $a \in \mathbb{N}_0$ ein additives Inverses $-a \in \mathbb{N}_0$ gibt.

(ii) $(\mathbb{Z}, +)$ ist eine abelsche Gruppe.

(iii) $(\mathbb{Z}^n, +)$ und $(\mathbb{Q}^n, +)$ sind abelsche Gruppen, wenn die Addition komponentenweite definiert ist. Dann ist $(0, ..., 0)$ das neutrale Element.

(iv) $(\mathbb{Z}, \cdot)$ ist keine Gruppe da es nicht zu jedem $a \in \mathbb{Z}$ ein multiplikatives Inverses $a^{-1} \in \mathbb{Z}$ gibt.

(v) $(\mathbb{Q} \setminus \{0\}, \cdot)$ ist eine abelsche Gruppe.

Beispiel 5.5 (Restklassen)**.** Betrachte die Menge $n\mathbb{Z} := \{n \cdot z \mid z \in \mathbb{Z}\}$. Definiere nun eine Relation $\sim_n$ auf $\mathbb{Z}$ durch $z_1 \sim_n z_2 :\Leftrightarrow z_1 - z_2 \in n\mathbb{Z}$. Es lässt sich leicht nachprüfen, dass $\sim_n$ eine Äquivalenzrelation ist. Wir bezeichnen die Äquivalenzklassen mit $\overline{z} := [z]$ für alle $z \in \mathbb{Z}$. Offensichtlich gibt es n Äquivalenzklassen, d. h. $\mathbb{Z}/n\mathbb{Z} := \{\overline{0}, ..., \overline{n-1}\}$. Wir definieren innere Verknüpfungen „$\cdot$“ und „$+$“ wie folgt:

$$\begin{aligned} \overline{a} + \overline{b} &= \overline{a+b} \\ \overline{a} \cdot \overline{b} &= \overline{a \cdot b} \end{aligned}$$

Man muss hierfür nachprüfen, dass das Ergebnis dieser Verknüpfungen nicht von der Wahl der Repräsentanten abhängt. Das kann der motivierte Leser als Übungsaufgabe gerne machen. Nun zum Gruppencharakter:

(i) Offenkundig ist $(\mathbb{Z}/n\mathbb{Z}, +)$ eine abelsche Gruppe.

(ii) $(\mathbb{Z}/n\mathbb{Z} \setminus \{0\}, \cdot)$ ist nicht immer eine Gruppe. So hat z. B. $\overline{2} \in \mathbb{Z}/4\mathbb{Z}$ kein Inverses, da $\overline{k} \cdot \overline{2} \neq \overline{1}$ für alle $\overline{k} \in \mathbb{Z}/4\mathbb{Z}$.

Satz 5.6. *Es sei $(G, *)$ Gruppe. Dann gilt:*

(i) *G besitzt genau ein neutrales Element.*

(ii) *Jedes $g \in G$ besitzt genau ein Inverses.*

Beweis.

(i) Seien $e_G, e'_G \in G$ neutrale Elemente. Dann gilt $e_G = e'_G * e_G = e'_G$

(ii) Seien g', g'' Inverse von g. Dann gilt $g' = g' * (g * g'') = (g' * g) * g'' = g''$

□

Definition 5.7. Seien $(G, *_G)$ und $(G', *'_G)$ zwei Gruppen. Eine Abbildung $\varphi : G \to G'$ heißt *Gruppenhomomorphismus*, falls für alle $g, g' \in G$ gilt:

$$\varphi(g *_G g') = \varphi(g) *'_G \varphi(g')$$

Falls φ bijektiv ist, so heißt φ *Gruppenisomorphismus*. Falls zwischen G und G' ein Isomorphismus existiert, so heißen G und G' *zueinander isomorph*, schreibe $G \cong G'$.

Definition 5.8. Ein *Körper* $(\mathbb{K}, +, \cdot)$ ist eine nichtleere Menge $\mathbb{K}$ zusammen mit zwei inneren Verknüpfungen

$$\begin{aligned} \text{add} &: \mathbb{K}^2 \to \mathbb{K}, (r_1, r_2) \mapsto r_1 + r_2 \\ \text{mult} &: \mathbb{K}^2 \to \mathbb{K}, (r_1, r_2) \mapsto r_1 \cdot r_2 \end{aligned}$$

und folgenden Eigenschaften:

(i) $(\mathbb{K}, +)$ ist eine abelsche Gruppe.

(ii) $(\mathbb{K} \setminus \{0\}, \cdot)$ ist eine abelsche Gruppe.

(iii) „$+$“ und „$\cdot$“ sind *distributiv*, d. h. für $r_1, r_2, r_3 \in \mathbb{K}$ gilt: $r_1 \cdot (r_2 + r_3) = (r_1 \cdot r_2) + (r_1 \cdot r_3)$

Beispiel 5.9.

(i) $(\mathbb{Z}, +, \cdot)$ ist kein Körper, da nicht zu jedem $a \in \mathbb{Z} \setminus \{0\}$ ein multiplikatives Inverses $a^{-1} \in \mathbb{Z}$ existiert.

(ii) $(\mathbb{Q}, +, \cdot)$ ist ein Körper.

(iii) $(\mathbb{Z}/n\mathbb{Z}, +, \cdot)$ ist genau dann ein Körper, wenn n prim ist.

(iv) $(\mathbb{Q}^n, +, \cdot)$ ist kein Körper, auch wenn beide Verknüpfungen punktweise definiert sind, da es nicht zu jedem Vektor ein komponentenmultiplikatives Inverses gibt.

Bemerkung 5.10. Sei $(\mathbb{K}, +, \cdot)$ ein Körper. Es wurden folgende Schreibweisen vereinbart:

(i) Werden keine Klammern gesetzt, so wird zunächst die multiplikative und dann die additive Verknüpfung ausgeführt.

(ii) Statt der Addition eines additiven Inversen wird eine *Subtraktion* notiert:

$$r_1 - r_2 := r_1 + (-r_2)$$

(iii) Statt der Multiplikation mit einem multiplikativen Inversen wird eine *Division* notiert:

$$\frac{r_1}{r_2} := r_1 \cdot r_2^{-1}$$

Satz 5.11. *Sei $(\mathbb{K}, +, \cdot)$ ein Körper. Dann gilt:*

(i) *Für alle $r \in \mathbb{K}$ gilt $0_\mathbb{K} \cdot r = 0_\mathbb{K} = r \cdot 0_\mathbb{K}$*
(ii) *Für alle $r \in \mathbb{K}$ gilt $(-1_\mathbb{K}) \cdot r = -r = r \cdot (-1_\mathbb{K})$*
(iii) *Für alle $r_1, r_2 \in \mathbb{K}$ mit $r_1 \cdot r_2 = 0$ gilt $r_1 = 0$ oder $r_2 = 0$.*

Beweis.

(i) Es gilt $0_\mathbb{K} \cdot r = (0_\mathbb{K} + 0_\mathbb{K}) \cdot r = 0_\mathbb{K} \cdot r + 0_\mathbb{K} \cdot r$ (Distributivgesetz). Addiert man auf beiden Seiten $-0_\mathbb{K} \cdot r$, so gilt $0_\mathbb{K} \cdot r = 0_{(\mathbb{K})}$. Analog für die rechtsseitige Anknüpfung.

(ii) Für alle $r \in \mathbb{K}$ gilt: $(-1_\mathbb{K}) \cdot r + r = (-1_\mathbb{K} + 1_\mathbb{K}) \cdot r = 0_\mathbb{K} \cdot r = 0_\mathbb{K}$. Also ist $(-1_\mathbb{K}) \cdot r = -r$. Analog für die rechtsseitige Anknüpfung.

(iii) Für $r_1 = 0_\mathbb{K}$ ist die Behauptung bereits gezeigt. Betrachte $r_1 \neq 0_\mathbb{K}$. Dann gibt es $r_1^{-1} \in \mathbb{K}$. Es gilt:

$$\begin{aligned} r_2 &= 1_\mathbb{K} \cdot r_2 \\ &= (r_1^{-1} \cdot r_1) \cdot r_2 \\ &= r_1^{-1} \cdot (r_1 \cdot r_2) \\ &= r_1^{-1} \cdot 0_\mathbb{K} \\ &= 0_\mathbb{K} \end{aligned}$$

□

Definition 5.12. Seien $(\mathbb{K}, +_\mathbb{K}, \cdot_\mathbb{K})$ und $(\mathbb{K}', +'_\mathbb{K}, \cdot'_\mathbb{K})$ zwei Körper. Eine Abbildung $\varphi : \mathbb{K} \to \mathbb{K}'$ heißt *Körperhomomorphismus*, falls $\varphi(1_\mathbb{K}) = 1_{\mathbb{K}'}$ und für alle $r, r' \in \mathbb{K}$ gilt:

$$\begin{aligned} \varphi(r +_\mathbb{K} r') &= \varphi(r) +'_\mathbb{K} \varphi(r') \\ \varphi(r \cdot_\mathbb{K} r') &= \varphi(r) \cdot'_\mathbb{K} \varphi(r') \end{aligned}$$

Falls φ bijektiv ist, so heißt φ *Körperisomorphismus*. Falls zwischen $\mathbb{K}$ und $\mathbb{K}'$ ein Isomorphismus existiert, so heißen $\mathbb{K}$ und $\mathbb{K}'$ *zueinander isomorph*, schreibe $\mathbb{K} \cong \mathbb{K}'$.

Definition 5.13. Ein Körper $(\mathbb{K}, +, \cdot)$ zusammen mit einer Vollordnung $\leq$ heißt *geordnet*, wenn folgende Eigenschaften gelten:

(V1) Für alle $r_1, r_2, r' \in \mathbb{K}$ mit $r_1 \leq r_2$ gilt $r_1 + r' \leq r_2 + r'$
(V2) Für alle $r_1, r_2, r' \in \mathbb{K}$ mit $r_1 \leq r_2$ und $r' \geq 0_\mathbb{K}$ gilt $r' \cdot r_1 \leq r' \cdot r_2$

Ferner definiere noch folgendes:

(i) $r \geq 0_\mathbb{K}$ heißt *nicht-negativ*, $r > 0_\mathbb{K}$ heißt *positiv*.
(ii) $r \leq 0_\mathbb{K}$ heißt *nicht-positiv*, $r < 0_\mathbb{K}$ heißt *negativ*.

Satz 5.14. *Sei $(\mathbb{K}, +, \cdot, \leq)$ ein geordneter Körper. Dann gilt für alle $r \in \mathbb{K}$: $r^2 \geq 0_\mathbb{K}$.*

Beweis. Für $r \geq 0$ ist $r \cdot r \geq r \cdot 0_\mathbb{K} = 0_\mathbb{K}$ (V2). Betrachte nur noch $r < 0_\mathbb{K}$. Es ist klar, dass für alle $r' \in \mathbb{K}$ die Äquivalenz $r' \geq 0 \Leftrightarrow -r' \leq 0$ (V1) gilt. Ferner wissen wir mit Satz 5.11 $(-1_\mathbb{K})^2 = 1_\mathbb{K}$. Folglich gilt:

$$\begin{aligned} r^2 &= (-1_\mathbb{K})^2 \cdot r^2 \\ &= ((-1_\mathbb{K}) \cdot r)^2 \\ &= (-r)^2 \\ &\geq 0_\mathbb{K} \end{aligned}$$

□

Satz 5.15. *Sei $(\mathbb{K}, +, \cdot, \leq)$ ein geordneter Körper und $r \in \mathbb{K}$. Dann sind folgende Aussagen äquivalent:*

(i) $r > 0_\mathbb{K}$
(ii) $r^{-1} > 0_\mathbb{K}$

Beweis.

(i) „⇒“:
Ang. $r > 0$ und $r^{-1} \leq 0$. Dann gilt nach V2: $r \cdot r^{-1} \leq r \cdot 0_\mathbb{K} = 0_\mathbb{K}$. Folglich ist $1_\mathbb{K} \leq 0_\mathbb{K}$. Ferner ist aber $1_\mathbb{K} = (-1_\mathbb{K})^2 \geq 0_\mathbb{K}$. Durch die Antisymmetrie der Ordnungsrelation wäre demnach $1_\mathbb{K} = 0_\mathbb{K}$, was einen Widerspruch darstellt.

(ii) „⇐“:
Sei $r^{-1} > 0$. Dann ist wegen der Gültigkeit der Hinrichtung: $(r^{-1})^{-1} > 0$, also $r > 0$.

□

Definition 5.16. Sei $(\mathbb{K}, +, \cdot)$ ein Körper. Betrachte die Abbildung

$$\varphi_\mathbb{K} : \mathbb{N} \to \mathbb{K}, n \mapsto \underbrace{1_\mathbb{K} + \cdots + 1_\mathbb{K}}_{n \text{ mal}}$$

Falls $0_\mathbb{K} \notin \varphi_\mathbb{K}(\mathbb{N})$, so schreibe $\mathrm{char}(\mathbb{K}) = 0$, andernfalls:

$$\mathrm{char}(\mathbb{K}) = \min(\varphi_\mathbb{K}^{-1}(0_\mathbb{K}))$$

Dann heißt $\mathrm{char}(\mathbb{K})$ *Charakteristik* des Körpers $\mathbb{K}$.

Satz 5.17. *Ein geordneter Körper hat Charakteristik 0.*

Beweis. Genau dann, wenn ein geordneter Körper $(\mathbb{K}, +, \cdot, \leq)$ Charakteristik 0 hat, gilt für alle $n \in \mathbb{N}$: $\varphi_\mathbb{K}(n) > 0_\mathbb{K}$. Letzteres zeigen wir über Induktion:

(i) *Induktionsanfang*
$n = 1$ ist das Einselement. Folglich ist $n_\mathbb{K} = 1 \neq 0$.

(ii) *Induktionsvoraussetzung*
Für ein beliebiges $n \in \mathbb{N}$ gelte $(n-1)_\mathbb{K} > 0_\mathbb{K}$

(iii) *Induktionsschritt*
Es gilt: $n_\mathbb{K} = (n-1)_\mathbb{K} + 1_\mathbb{K} \geq (n-1)_\mathbb{K} > 0_\mathbb{K}$

□

[1] http://fkranhold.de/?p=/Universit%C3%A4t/WS_2012-13/Analysis_1 (abgerufen am: 20.11.2013, 21:02)

Technik

Abfliegen von Wegpunkten ohne absolutes Ortungssystem durch einen Quadrokopter

von Lukas Heimann, Arjun Sarin (Gastbeitrag)

Sehr geehrter Leser, im Winter 2014 haben wir eine „Jugend forscht“-Arbeit mit dem etwas sperrigen Titel *Abfliegen von Wegpunkten ohne absolutes Ortungssystem durch einen Quadrokopter* veröffentlicht. Ziel war es, das Fluggerät, das wir liebevoll „Bernie“ getauft haben, autonom Wegpunkte in Innenräumen abfliegen zu lassen, vorerst nur Parabelflüge.

Teil 1 ist in der letzten Ausgabe des Neologismus erschienen, lesen Sie dieses Mal: Teil 2 – über Software, ermüdende Tests und ein Fazit.

Software

Das MultiWii-Projekt

Damit der Flightcontroller den Quadrokopter korrekt ansteuern kann, benötigt er geeignete Software. Diese liegt in dem Open-Source-Projekt *MultiWii* bereits vor.

Ursprünglich auf der Hardware der *Nintendo®-Wii™*-Fernbedienung aufbauend, ist die Software inzwischen in der Lage, beinahe jede Art von Multikopter anzusteuern – habe er nun 1, 2, 3, 4 oder 8 Propeller. Dabei deckt die Funktionalität bereits sehr vielfältige Probleme ab: Neben der simplen Ansteuerung des Multikopters per Fernsteuerung werden erweiterte Einstellungen auch über ein eigenes Computerprogramm zugänglich gemacht. Des Weiteren werden unterschiedliche Flugmodi für stabilen Flug und Kunstflug, Unterstützung für viele Sensoren und sogar ein Wegpunkte-System via GPS zur Verfügung gestellt.

Für unser Projekt haben wir das zu Projektbeginn aktuelle MultiWii 2.2 verwendet.

Die Stabilisierung

Ein Multikopter wird über Regelkreisläufe stabilisiert, wie sie aus der Regelungstechnik bekannt sind: Es liegen ein Messwert (Ist-Wert) und ein Soll-Wert vor, was eine Abweichung e verursacht. Diese soll kompensiert werden, wozu ein entsprechend starkes Ausgangssignal generiert werden muss. Das MultiWii verwendet PID-Regler, weil diese aufgrund ihrer genauen und schnellen Reaktion am besten geeignet sind. Sie bestehen aus den folgenden drei Teilreglern:

Der *P-Regler* (für „proportional“) sorgt für eine Korrektur, deren Stärke proportional zu Abweichung vom Sollzustand ist:

```
PTerm:=P*e;
```

Der P-Regler stellt die einfachste und schnellste Art der Korrektur dar, jeder Abweichung wird sofort entgegengewirkt.

Der *I-Regler* (für „Integral“) sorgt für eine Korrektur, deren Stärke zunimmt, je länger die Abweichung vom Sollzustand besteht:

```
e_sum:=e_sum+e;
ITerm:=I*e_sum;
```

Durch das Aufsummieren reagiert der I-Regler sehr langsam. Da sich seine Stärke jedoch erst wieder verringert, wenn der Ist-Wert erreicht wird, führt er zu einer vollständigen Auflösung der Abweichung (im Gegensatz zum P-Regler, bei dem sich der Ist-Wert nur asymptotisch dem Soll-Wert annähert).

Zusätzlich wird noch ein *D-Anteil* (für „Differential“) verrechnet, der die Änderungsrate der Abweichung betrachtet:

```
DTerm:=D*(e-e_old);
e_old:=e;
```

Der D-Anteil selbst stellt keinen Regler dar, da er die Abweichung selbst nicht korrigiert, sondern nur ihre Änderungsrate betrachtet und sie so anpassen kann.

Im letzten Schritt werden die drei Terme aufaddiert, wobei der D-Anteil abgezogen wird, da er zu schnelle Änderungen abfedern soll:

```
PIDTerm:=PTerm+ITerm-DTerm;
```

Dieser PID-Term wird im MultiWii für alle drei Raumachsen berechnet. Da jeder Motor Einfluss auf alle Raumachsen hat, werden die einzelnen PID-Terme verrechnet und anschließend auf die Motoren geschrieben.

Die oben eingeführten Proportionalitätsfaktoren P, I und D sind für alle Achsen einstellbar. Hierbei ist darauf zu achten, dass der Multikopter seine Soll-Position in möglichst kurzer Zeit erreicht und hält, ohne dabei zu sehr überzuschwingen und in der Luft zu flattern.

Das serielle Protokoll

Bereits zur Kommunikation mit dem Computer über die mitgelieferte Oberfläche verwendet das MultiWii ein eigenes Protokoll zur Datenübertragung, das verschiedene Verfahren zur Sicherstellung korrekter Übertragung verwendet. Dabei wurde es so implementiert, das es auf jedem seriellen Anschluss des Flightcontrollers verfügbar ist, ohne etwaige andere Nutzung von diesem

zu stören. Dazu wird folgender Kniff verwendet: Daten werden nur nach einer korrekten Anfrage im Sinne des Protokolls gesendet. Anfrage und Antwort sind sich sehr ähnlich und nach folgendem Schema aufgebaut:

(i) Nachrichtenkopf (*header*)
Ein spezifischer Header stellt sicher, dass es sich tatsächlich um eine Übertragung aus dem Protokoll handelt: Anfragen beginnen mit `$M<`, Antworten mit `$M>`, Fehlermeldungen mit `$M!`

(ii) Nachrichtenlänge (*length*)
Das Protokoll schickt jedes Zeichen (auch die des Headers) als Byte, sprich 8 Bit, die nach dem ASCII-Code als Buchstaben bzw. Zeichen, oder eben als Zahlen gedeutet werden können. Da der Nachrichteninhalt eine variable Länge (in Byte) haben kann, wird diese hier angegeben, damit dem Empfänger klar ist, wie lange er „zuhören“ muss.

(iii) Nachrichtentyp (*message type*)
Der Typ gibt an, auf welche Funktionalität zugegriffen werden soll. Das MultiWii weist dazu verschiedenen Typen verschiedene Funktionen zu. So wird auf Typ 100 zum Beispiel über den allgemeinen Zstand des Multikopters Auskunft angefragt bzw. gegeben.

(iv) Nachrichteninhalt (*payload*)
Der Inhalt der Nachricht. Die Reihenfolge der Informationen ist je nach Nachrichtentyp vorgegeben und muss der vorher angegebenen Länge entsprechen.

(v) Prüfsumme (*CRC*)
Zum Schluss wird noch eine Prüfsumme übertragen, damit eine fehlerhafte Übertragung zum Beispiel durch Störung des Signals ausgeschlossen werden kann. Ohne zur Nachricht passende Prüfsumme wird die Nachricht auf Seiten des Multikopters ignoriert. Die Prüfsumme wird generiert, indem Nachrichtenlänge, -typ und -inhalt nacheinander Byte für Byte logisch mit einem bitweisen XOR verknüpft werden.

Die hinzugefügte Funktionalität für unser Projekt

Für unsere Zwecke mussten wir an drei wesentlichen Stellen Code hinzufügen bzw. verändern:

(i) Wegpunkte und Fernsteuerungsroutine
Da wir keine normale Fernsteuerung verwenden, wurde dieser Code auskommentiert. Stattdessen haben wir eine Routine eingefügt, die aus den eingegebenen Wegpunkten Fernsteuerungseingaben errechnet, mit der das MultiWii den Quadrokopter ansteuert. Dazu haben wir für jede Raumachse einen Proportionalitätsfaktor eingeführt, der den Abstand vom letzten zum nächsten Wegpunkt als Stärke der auszuführenden Bewegung für eine bestimmte Zeit auf die entsprechende Achse der Fernbedienungseingabe schreibt.

(ii) Abstandssensor (Sonar)
Die Ansteuerung und insbesondere die Auswertung des Sonars war bislang im MultiWii nur ansatzweise implementiert. Der Code zur Auswertung der Messergebnisse musste komplett selbst geschrieben werden.

(iii) Serielles Protokoll
Das bereits implementierte serielle Protokoll stellt bereits viele nützliche Funktionen bereit, dennoch haben wir im Zuge der Entwicklung ein paar neue Nachrichtentypen für unseren Bedarf hinzugefügt, darunter Lese- und Schreibroutinen für Wegpunkte, den Flugstatus des Quadrokopters und die Proportionalitätsfaktoren, die bereits unter Punkt 1 erwähnt wurden.

Zusätzlich mussten wir ein eigenes Computerprogramm schreiben, das die von uns hinzugefügten und für uns relevanten Funktionen anzeigen, verarbeiten und senden kann. Dazu haben wir die freie Programmierumgebung *Lazarus* verwendet.

Bau und Entwicklung des Quadrokopters

Der Zusammenbau

Nachdem alle Bauteile eingetroffen waren, wagten wir uns direkt an den Bau des Quadrokopters. Orientiert haben wir uns dabei an einer Bauanleitung aus der *c't Hardware Hacks 03/13*, dank derer keine größeren Probleme beim Zusammenbauen auftraten. Das Bespielen des Flightcontrollers mit der MultiWii-Software war schnell geschehen, so dass danach alle Funktionen erfolgreich ein erstes Mal getestet werden konnten. Wir brachten am Quadrokopter noch eine kabelgebundene Fernsteuerung mit einem Notschalter an. Ausgelöst kappt dieser sofort die gesamte Stromverbindung des Flightcontrollers und schaltete dadurch alle Motoren gleichzeitig ab. So sollte er für uns als letzte Sicherung dienen, falls die Software komplett aus unserer Kontrolle geraten sollte.

Bei der Installation des Ultraschallsensors wurde es dann zum ersten Mal knifflig. Eine Routine für gewisse Sensoren ist standardmäßig im MultiWii-Projekt integriert, jedoch stellte sich diese zumindest für unseren Sensor als nicht funktionsfähig heraus. Dementsprechend mussten wir einen anderen Weg gehen und haben mit Hilfe der Arduino-Bibliothek *Wire*, die den I2C-Bus

unterstützt, eine eigene Ansteuerung implementiert. Die Auswertung einer Messung wird vom Prozessor unseres Sensorchips übernommen, so dass unser Code nur die Messwerte abrufen muss.

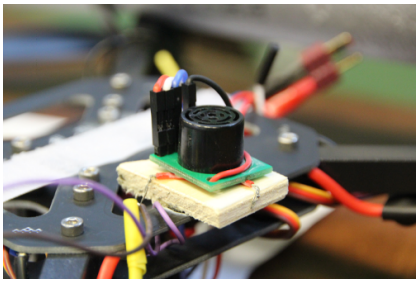


Abb. 2.1: Der Ultraschallsensor

Das Einrichten des Bluetooth-Adapters war kein Problem. Zwar ist dieser nicht im MultiWii vorgesehen, kann jedoch durch einfache Anpassung der Geschwindigkeit eines seriellen Busses angeschlossen werden, so dass wir anschließend mit der Entwicklung der Benutzeroberfläche fortfahren konnten.

Erfreulicherweise verliefen auch die ersten Tests der Datenübertragung von Laptop mit unserem Computerprogramm und dem MultiWii erfolgreich. Während des Fluges stellte sich aber die Bedienung mit Hilfe einer Maus als zu langsam heraus, weswegen wir wichtige Buttons der GUI mit Hardware-Shortcuts ausstatteten. Dadurch war eine schnelle Bedienung auch ohne Maus möglich.

Mit fortschreitender Entwicklung des Computerprogramms wurden allerdings auch die Befehlssequenzen immer komplexer. In bestimmten Fällen kam es sogar zu Überlagerungen verschiedener Befehle und Nachrichten wurden vom Computer schneller gesendet als sie vom MultiWii verarbeitet werden konnten. Dementsprechend ging die Rückantwort des MultiWii manchmal verloren und das Programm hing sich auf. Eine einfache Lösung haben wir mit dem Einrichten einer Sendeschleife implementiert. Alle zu schickenden Nachrichten des Computerprogramms werden in eine globale Befehlswarteschlange geschrieben. Diese wird gesondert von einer zeitgesteuerten Schleife abgearbeitet und neue Nachrichten werden erst nach Erhalt einer validen Rückantwort gesendet oder bei Übertragungsfehlern nach dreimaligem Senden bzw. nach einem gewissen Timeout.

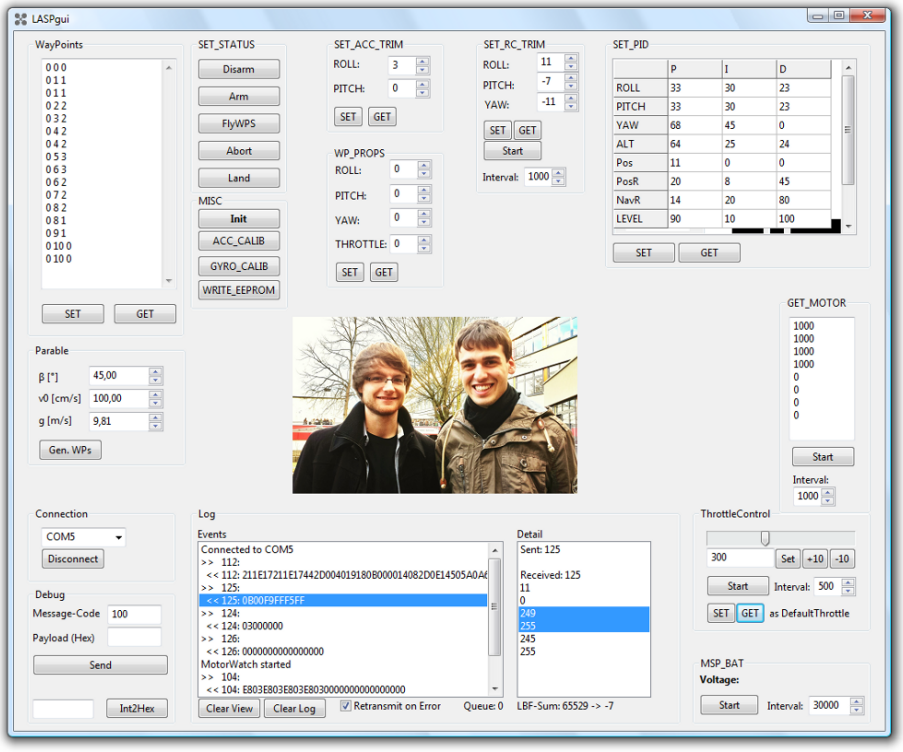


Abb. 2.2: Computerprogramm (LASPgui) zur Steuerung des Quadrokopters

Kalibrieren und Trimmen des Quadrokopters

Nachdem der Quadrokopter fertig aufgebaut und die Software sowohl auf dem Computer als auch dem NanoWii fertig entwickelt und aufgespielt war, ging es ans Kalibrieren und Trimmen des Quadrokopters, damit sich dieser stabil in der Luft hält. Das Kalibrieren eicht alle Sensoren auf eine Nulllage; diesen nur Sekunden dauernden Prozess konnten wir über unser Programm aus der Ferne starten. Komplizierter ist das Trimmen. Dabei müssen die verschiedensten Variablen, die bei der Stabilisierung des Quadrokopters eine Rolle spielen, eingestellt werden, was leider hauptsächlich über Ausprobieren funktioniert.

Zuerst haben wir versucht, die PID-Werte korrekt einzustellen. Dabei kann man für stabilen Flug grob folgender Regel folgen:

(i) P-Wert so lange erhöhen, bis der Quadrokopter zu schwingen beginnt; dann wieder leicht erniedrigen. Dies führt zu einer möglichst starken Ausgleichsreaktion.

(ii) I-Wert so lange erniedrigen, bis der Quadrokopter nicht mehr stabil fliegt; dann wieder leicht erhöhen. Dies erhöht die Reaktionsgeschwindigkeit, da Abweichungen in der Vergangenheit weniger stark beachtet werden.

(iii) D-Wert erhöhen, bis der Flug schön gleitend ist. Hohe D-Werte führen zu einer stärkeren Kompensation von „hastigen“ Ausgleichbewegungen.

Beim Versuch, die Werte einzustellen, stellte sich die Frage, wie man den Quadrokopter ohne Fernbedienung denn überhaupt initial starten kann. So musste das Programm also noch um die Funktionalität ergänzt werden, dem Quadrokopter einen festen Wert für *Throttle* (Schubkraft), zuzuweisen. So wollten wir den Quadrokopter auf eine bestimmte Höhe bringen, auf der er dann stabil die Position halten sollte.

Bei unseren ersten Versuchen hielten wir den Quadrokopter noch dicht am Boden, da wir auf Grund unserer eingeschränkten Kontrollmöglichkeiten kein großes Risiko eingehen wollten. Wenn wir allerdings dann in die Luft abgehoben sind, war sehr schnell ein Wegkippen zu beobachten. Das Problem war der I-Term: Bei laufenden Motoren auf dem Boden hat sich eine minimale Abweichung vom Sollzustand über längere Zeit aufsummiert; Resultat war direkt nach dem Abheben eine starke Gegenreaktion. Die Lösung dafür war, den Abstandssensor dazu einzusetzen, die für den I-Term aufsummierten Abweichungen auf 0 zu setzen, solange der Quadrokopter nicht eine bestimmte Höhe über Grund erreicht hatte.

Trotzdem konnte der Quadrokopter die Position in der Luft noch nicht vollends halten, solange nur die PID-Werte verändert wurden. Grund dafür sind einige andere, bisher noch nicht beachtete Faktoren gewesen: So kommt es vor, dass sich der Quadrokopter nach der Kalibrierung der Sensoren immer noch in einer leicht schiefen Lage zu befinden glaubt, was, wenn niemand gegensteuert, zu dem von uns beobachteten Wegdriften führt. Also musste nach jeder Kalibrierung dieser sogenannte *angleTrim* ebenfalls eingestellt werden, um diese vermutete Schieflage zu kompensieren.

Immer noch blieb der Drift bestehen. Weitere Überlegungen und eine Rückfrage bei der Community, die das MultiWii entwickelt, brachten uns auf die Fährte, dass zu überlegen ist, wie Quadrokopter im Normalfall stabilisiert werden, wenn man eine Fernbedienung benutzt: Man trimmt diese. Der mittlere Wert, den eine Fernbedienung an den Flightcontroller sendet wird auf 1500 gesetzt und bewegt sich dann sehr grob im Bereich zwischen 1000 und 2000. Wenn ein Quadrokopter abdriftet, ist die normale Reaktion, diesen mittleren Wert auf Seite der Fernbedienung zu korrigieren. Nun mussten wir in unsere Software also eine Routine einbauen, die dieses Nachstellen der Fernbedienung auf dem Quadrokopter simuliert. Leider brachte auch dieses Vorgehen nicht die erhoffte Besserung: Beim Start und innerhalb der ersten Sekunden bleibt der Quadrokopter stabil, dann jedoch bricht er plötzlich nach einer Seite aus und fliegt davon.

Die genauen Ursachen dafür konnten wir noch nicht ermitteln, wir vermuten allerdings ein Zusammenspiel aus einem Luftzug und ungenauen Sensordaten, was leider bedeutet, dass exakte Stabilisierung ohne absolutes Ortungssystem auch in geschlossenen Räumen in der Praxis nicht oder nur sehr bedingt möglich ist. Bei nicht-autonomen Multikoptern werden diese Probleme manuell ausgeglichen.

Das Abfliegen von Wegpunkten

Trotz dieser Ungenauigkeiten haben wir uns entschieden, unser Programm zum Abfliegen der Wegpunkte zu testen. Dazu haben wir die in der Einleitung erwähnte Routine verwendet, die uns als Teststrecke die Wegpunkte eines Parabelflugs generiert. Die Proportionalitätsfaktoren für die Wegpunkte müssen ebenfalls erstmalig durch abschätzen und ausprobieren eingestellt werden.

Ein Problem, das wir dabei ursprünglich nicht beachtet haben, ist die Zeit, die der Quadrokopter bis zum nächsten Wegpunkt brauchen soll. Diese mussten wir auf Grund unseres Ansteuerungssystems festlegen, wobei uns der Spagat zwischen Ungenauigkeit beim Parabelflug durch zu wenige Wegpunkte und einer zu großen Menge an Wegpunkten, was einerseits die Übertragung verlängert und andererseits Speicherprobleme auf dem Quadrokopter auslösen kann, vor eine besondere Herausforderung stellte.

Bei der korrekten Einstellung der Proportionalitätsfaktoren ist zu beachten, dass diese von der oben genannten Aktualisierungszeit abhängig ist. Zudem musste der Wert für die Schubkraft, die benötigt wird, den Quadrokopter einfach auf stabiler Höhe zu halten, eingestellt werden, damit der Parabelflug nicht zu flach wird und der Quadrokopter nicht den Boden erreicht, bevor er dies eigentlich sollte.

Dennoch ist uns bei unseren Testflügen ein guter Parabelflug gelungen, dessen Genauigkeit wir in Zukunft durch weitere Tests und bessere Einstellung der vielen Variablen des Programms zu verbessern hoffen.

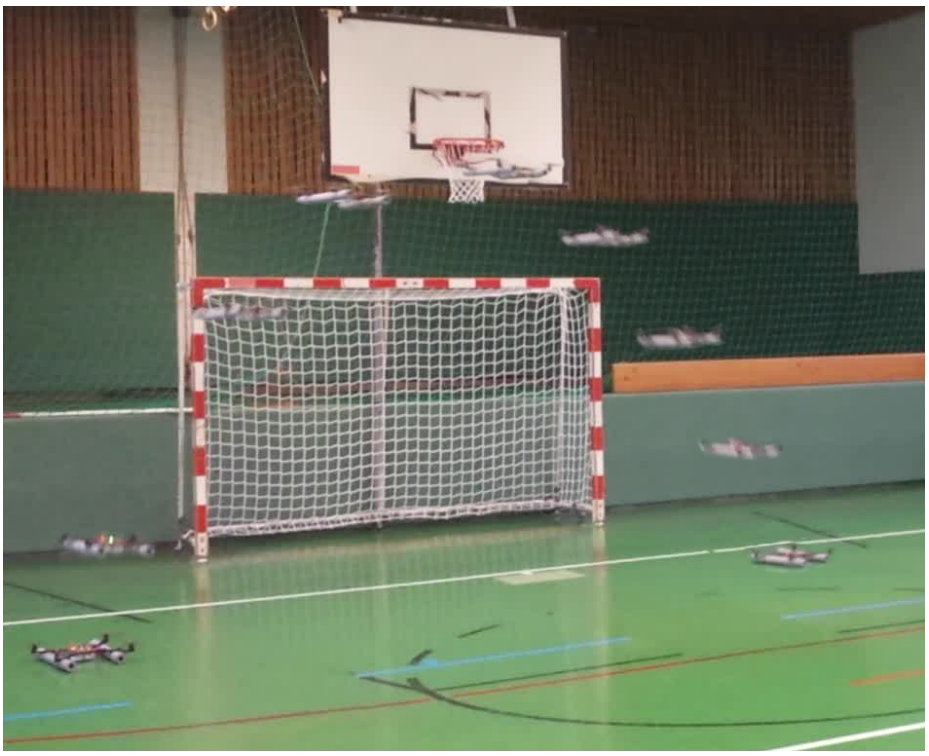


Abb. 2.3: Abflug einer Wurfparabel, Bilder aus Filmsequenz

Fazit

Multikopter sind cool! Das ist keine Frage. Neueste Veröffentlichungen großer Firmen wie der *Deutschen Post*-Tochter *DHL* oder *Amazon*, die Forschungsprojekte unterhalten, um in Zukunft Pakete schneller mit kleinen eigenständigen Drohnen liefern zu können, zeigen die Aktualität solcher Technik. Sogar der erste bemannte Flug wurde Ende 2013 mit einem von drei deutschen Technikern der Firma *e-volo* entwickelten elektrischen Multikopter absolviert. Andere Forschungsinstitutionen entwickeln schon eigenständig intelligente Quadrokopter, die für einen Menschen zwar einfache, für eine Maschine jedoch recht komplexe Dinge wie zum Beispiel das Werfen und Auffangen eines Stabes beherrschen. Sogar die Interaktion und Zusammenarbeit mit anderen Fluggeräten ist schon auf einem äußerst erstaunlichen Niveau angelangt.

Auch das Interesse im Modellbaubereich an Multikoptern wird immer größer, da fertige Softwareprojekte wie MultiWii den Einstieg auch für Anfänger mit sehr wenigen Kenntnissen stark vereinfachen. Fortgeschrittenen Entwicklern mit etwas mehr Erfahrung wird außerdem eine Plattform geboten, um ihre Programmierkünste an einem praktischen Projekt auszutesten.

Eine populäre Disziplin für Fluggeräte jeglicher Art ist es, immer wieder festgelegte Routen so genau wir möglich abfliegen zu können. Doch dies war in geschlossenen Räumen kostengünstig und einfach bisher nicht realisierbar. Diesen Umstand zu verändern war unsere Absicht, als wir mit dem Bau unseres Quadrokopters begonnen haben. Wir haben einen Weg gefunden, den Quadrokopter selbstständig fliegen zu lassen, Wegpunkte abzufliegen und anschließend auch wieder zu landen. Dabei konnten wir eine vollständige Steuerung mit Hilfe eines Computers über eine grafische Benutzeroberfläche installieren. Während des Fluges kann der Quadrokopter auch komplett autonom agieren und es werden keine weiteren Befehle von einem externen Computer benötigt. Dabei können wir auf GPS oder externe Kameras verzichten und arbeiten nur mit Hilfe von drei Sensoren. Die Stabilisierung in der Luft und das Halten einer bestimmten Position nur mit diesen Sensoren ist allerdings nur mit einer gewissen Ungenauigkeit zu realisieren. Auch in geschlossenen Räumen kann es zu unvorhersehbaren Luftzügen kommen, die den Quadrokopter abdriften lassen. Außerdem verfügen die Sensoren nur über eine begrenzte Genauigkeiten und selbst bei korrekter Kalibrierung, weichen die Messwerte nach einer gewissen Flugzeit von den tatsächlichen Werten teilweise stark ab. Nicht zu vernachlässigen sind schließlich die Gehäuseschwingungen, die perfekte Sensordaten unmöglich machen. Auch wenn die Sensorik durch Gummiplatten vom Gehäuse entkoppelt wurde, beeinflussen die Bewegungen der Motoren praktisch die gemessenen Daten.

Trotzdem lässt sich zusammenfas-

sen, dass die Forschung mit Quadrokoptern ein großes Potential hat und bei ausgefeilter Technik in Zukunft auch unseren Alltag beeinflussen könnte. Die Entwicklungen sind rasant, auch weil sich viele Menschen im Hobbybereich ernsthaft mit der Weiterentwicklung von Multikoptern beschäftigen.

Danksagung

Wir möchten uns beim Kurfürst-Salentin-Gymnasium Andernach in Person von Herrn Walter Reis für die finanzielle Unterstützung bedanken, ohne die unser Projekt nicht hätte realisiert werden können.

[1] http://www.multiwii.com (abgerufen am: 02. 01. 2014)

[2] **Brunnes, Lisa.** *Erster bemannter Flug mit einem elektrischen Multikopter*. netzwelt.de/news/89264-kurzfilm-erster-bemannter-flug-elektrischen-multikopter.html (abgerufen am 28. 12. 2013)

[3] **Waste.** *Regelungstechnik*. rn-wissen.de/index.php/Regelungstechnik (abgerufen am 02. 01. 2014)

[4] **Quandt, Roland.** *DHL: Erster Testflug mit neuer Paket-Lieferdrohne*. winfuture.de/news,79258.html (abgerufen am 28. 12. 2013)

[5] **Brescianini, Dario; Hehn, Markus; D'Andrea, Raffaello.** (ETH Zürich) *The astounding athletic power of quadcopters*. youtube.com/watch?v=w2itwFJCgFQ (abgerufen am 05.01.2014)

[6] **Brescianini, Dario; Hehn, Markus; D'Andrea, Raffaello.** (ETH Zürich) *Quadrocopter Pole Acrobatics*. youtube.com/watch?v=pp89tTDxXuI (abgerufen am 05. 01. 2014)

[7] **Wegner, Jochen.** (Hrsg.) *Amazon will Ware per Mini-Drohne liefern*, erschienen auf *ZEIT online*, zeit.de/wirtschaft/2013-12/amazon-will-ware-per-drohne-liefern (abgerufen am 28. 12. 2013)

Langzeittestbericht zur smarten Uhr

von Marcel Hörz

Es wurde mit Sicherheit schon sehr viel über sie geschrieben und sie wurden wohl auch schon sehr oft getestet: Die neue Generation der Armbanduhr – die *Smartwatch*. Allerdings stehen in diesen Texten meist nur technische Daten oder Dinge, die einen auf dem ersten Blick begeistern. Ich allerdings möchte zeigen, wie sich die Uhr nach einem knappen Jahr bei mir im Alltag bewährt hat.

Viele Leute, denen ich begegne, fragen mich bzw. erklären mir Dinge wie „Was willst du denn damit?“, „Geldverschwendung!“ und „Braucht man das überhaupt?“. Nun, eigentlich braucht man sie nicht; ich bin auch ohne Uhr gut zurecht gekommen. Ich habe die Uhr aber jetzt schon seit Mai letzten Jahres und ich muss ehrlich sagen: Ich will sie nicht missen. Sie ist eine echte Erleichterung in einem Leben mit einem Smartphone. Zähle ich Beispiele auf, was meine Uhr (*Sony SmartWatch MN*) alles kann, so wird gerne darüber gelacht. Das kann ich erstmal gut verstehen, denn auf dem ersten Blick wirken sie wie kleine Spielereien. Okay, es sind ja auch einige Spielchen (Schiebepuzzle, Sodoku, Snake usf.) darauf. Aber auch die App, um SMSen zu schreiben, wirkt erstmal wie zu belächeln. Sogar im Internet kann man mit dem Displaychen browsen. Auch das wirkt zunächst lächerlich.

Wie man merkt, sind es alleine keine großen und wahnsinnig aufregenden Neuerungen in der Technik.

Im Alltag allerdings ist das ziemlich nützlich. Während alle erst eine Hand frei machen, ihr Handy aus der (Hosen-)Tasche kramen und entsperren müssen, um eine SMS zu lesen, reicht mir ein Blick auf das Handgelenk. Beantwortet sind diese entweder duch vordefinierte Texte (Ausreichend für „Ja“, „Nein“, „Melde mich später.“ und dergleichen.) oder durch eigene Texteingabe. Diese ist jedoch etwas aufwändig, lohnt sich aber immernoch, wenn man mal etwas Kurzes schnell tippen möchte, weil entweder kein vordefinierter Text passt oder man grad einfach nicht sein Handy rausholen kann oder will oder es auf der Uhr trotzdem schneller geht. Auch Mails lassen sich auf diese Art lesen und beantworten.

Ist man gerade unterwegs, hört Musik und einem gefällt das Lied nicht, lässt sich auch das durch ein Wischen ändern. Früher musste ich dafür erst mein Handy in die Hand nehmen. Möchte man dann noch jemanden anrufen, dann sind das auf der Uhr drei Tipps und das Handy wählt. Auch hier bleibt es, dank Headset, in der Tasche.

Auch *Google Maps* gibt es auf der Uhr. Die kleine Karte ist auf dem ersten Blick etwas gewöhnungsbedürftig, da man sich nicht quer durch das Land schieben kann. Aber um mal kurz zu sehen, wo man grad ist, reicht sie völlig. In einer anderen App werden auch die Geo-Koordinaten und Geschwindigkeit (mit Zeit) angezeigt, was beim Sport dann doch sehr nützlich sein kann.

Mit einem Notifier kann man sich sämtliche Meldungen, die auch in der Titelleiste des Handys eingehen, weiter auf die Uhr schicken lassen. So bekommt man, wenn das Handy lautlos ist, doch noch alles mit. Die App, von der ich in meiner Schulzeit wohl am meisten Gebrauch gemacht habe, ist die, mit der man den Lautstärkemodus des Handys ändern kann. Auch das geht sehr schnell und das Handy bleibt in der Tasche (oder lässt sich noch vor den Augen der Lehrer in den Lautlosmodus wechseln, ohne, dass sie etwas mitbekommen).

Wirklich interessant ist da noch die App, mit der sich Photos machen lassen. Zum einen als kleiner Spion in der Hosentasche (Wenn man

verdeckt Bilder machen möchte), zum anderen erspart man sich den Selbstauslöser. Man stellt die Kamera bzw. das Handy da ab, wo man es haben möchte, und kann dann auch sich selbst in Ruhe positionieren. Auf der Uhr sieht man auch das aktuelle Bild der Kamera. Mit einem Tipp lässt sich das Photo dann auch schießen.

Leider ist meine Uhr nicht immer schnell. Gerade die Bedienung bei schlechter Bluetooth-Verbindung wird schnell zur Geduldsprobe. Dieses Problem ist aber bei einigen Uhren heute schon gelöst, da teilweise sogar ganze Androids (*Android Wear* aber auch „echte“ Smartphone-Androiden) eigenständig auf der Uhr laufen. Von *Simvalley* gibt es sogar eine Uhr, in der man eine SIM-Karte einlegen kann und hat dann ein richtiges Handy am Arm.

Viele haben Bedenken, dass die Uhr aufgrund der Bluetooth-Verbindung das Akku des Handys leersaugen würde. An sich erst mal kein falscher Gedanke. Habe ich die Uhr mit dem Handy verbunden, trage die Uhr ungenutzt mit mir rum und bediene alles mit dem Handy, ist die Uhr wirklich an der Durchhaltedauer des Handy-Akkus bemerkbar. Normalerweise hält er bei durchschnittlicher Benutzung ca. einen Tag. Mit Bluetooth-Verbindung nur noch einen dreiviertel Tag. Was viele aber vernachlässigen, ist, dass man mit der Uhr deutlich seltener das Display des Handys einschaltet. Und das macht dann doch deutlich etwas aus, denn der Akku hält länger durch. Ab und an kann es sogar sein, dass ich mein Handy erst zwei Tage später aufladen muss.

Für viele problematischer ist jedoch, dass auch die Uhr aufgeladen werden muss: „Tja, und ich muss meine Uhr nicht aufladen. Meine hält mehrere Jahre!“, ist mir schon zu Ohren gekommen. Dazu sage ich nur all zu gerne: „Dafür kannst du aber auf Deiner Uhr keine SMSen schreiben!“ Es ist wohl allen klar: Eine Smartwatch frisst Strom. Entgegen allen Befürchtungen hält sie bei mittelmäßigem Gebrauch (nachts aus) aber gut fünf bis sechs Tage durch. Nur bei häufiger Verwendung kann man sie auch mal nach zwei, drei Tagen wieder aufladen. Das sind wirklich akzeptable Zeiten. Und hat man sie mal vergessen, aufzuladen, merkt man, dass sie doch wirklich nützlich ist.

Preislich fangen sie bei 20€ an und gehen bis in einen vierstelligen Bereich. Und selbst die Uhr für 20€ (*Sony LiveView*) ist nicht schlecht. Da sie aber nicht überall eine Touchfläche hat, ist man in der Bedienung etwas eingeschränkt. Im Groben sind jedoch fast alle oder zumindest von der Funktion her vergleichbare Apps der Sony SmartWatch auch für die LiveView erhältlich. Man sieht also: Man braucht nicht zwingend die relativ teure Uhr, die in letzter Zeit stark beworben wurde. Das Preis-Leistungsverhältnis ist bei der LiveView echt sehr gut und daher ist die Frage nach dem Geld nur eine nach dem, was man dafür haben möchte.

Alles in allem bin ich sehr zufrieden mit der Uhr (und dem Uhrenangebot). Es ist zwar (noch) nichts Weltbewegendes, aber eine sehr große Erleichterung im Alltag!

„Der GAU für Web-Verschlüsselung“

Über den „Heartbleed“-Bug in OpenSSL

von Michael Thies

Es kommt relativ selten vor, dass Fachthemen aus dem Bereich der Computertechnik so gesellschaftlich relevant sind, dass sie es bis in die 20Uhr-Nachrichten der Tagesschau schaffen. In diesem April war das gleich zweimal der Fall. Zum einen war da das Supportende von Windows XP, mit dem nun alle, die noch einen Computer mit diesem Betriebssystem betreiben, nach einer alternativen Lösung suchen sollten. Die c’t hat bereits vor vier Monaten in einem wunderbaren Editorial[1] zusammengefasst, warum.

Und zum anderen war da der „Heartbleed“ getaufte Bug in OpenSSL, über den ich im Folgenden mehr erklären möchte.

OpenSSL ist eine freie (unter einer OpenSource-Lizenz veröffentlichte) Software-Sammlung, die von vielen Programmen zum Aufbau von verschlüsselten Verbindungen verwendet wird. Diese SSL-Verbindungen (*SecureSocketLayer* oder auch TLS – *TransportLayerSecurity*) werden quasi immer verwendet, wenn Daten vertraulich über das Internet oder ein anderes Netzwerk übertragen werden müssen. Sie garantieren z.B. beim Online-Banking, beim E-Mail-Abruf und beim Login oder Bestellvorgang auf vielen Internetseiten, dass ein Unbeteiligter, selbst wenn er alle übertragenen Daten mitgeschnitten hat, keine Kontodaten, Mailinhalte oder Passwörter herausfinden kann. Und fast immer wird zum Aufbau dieser Verbindungen das OpenSSL-Paket verwendet.

Genau in dieser Software wurde Mitte April ein Programmierfehler gefunden, der eine schwere Sicherheitslücke darstellte. Der Fehler befand sich in der sogenannten *Heartbeat*-Funktion, die es den beiden Partnern, zwischen denen die verschlüsselte Verbindung besteht, ermöglicht, zu überprüfen, ob die Verbindung noch intakt ist. Da der Feh-

ler das Auslesen großer Datenmengen aus dem eigentlich geschützten Speicher des Servers ermöglicht, wurde er von den Entwicklern „Heartbleed“ getauft (dt. etwa „Herzbluten“).

Der Fehler

Bei der Heartbeat-Funktion schickt einer der Verbindungspartner einige beliebige Daten sowie deren Länge an den anderen Partner. Dieser schickt zur Bestätigung, dass die Verbindung noch funktioniert, die gleichen Daten zurück. Leider hat OpenSSL Version 1.0.1 vor dem Antworten nicht überprüft, ob die angegebene Länge auch der tatsächlichen Datenmenge entspricht. Wie sich das sehr einfach für einen Angriff ausnutzen lässt, zeigt der folgende XKCD-Cartoon anschaulich:

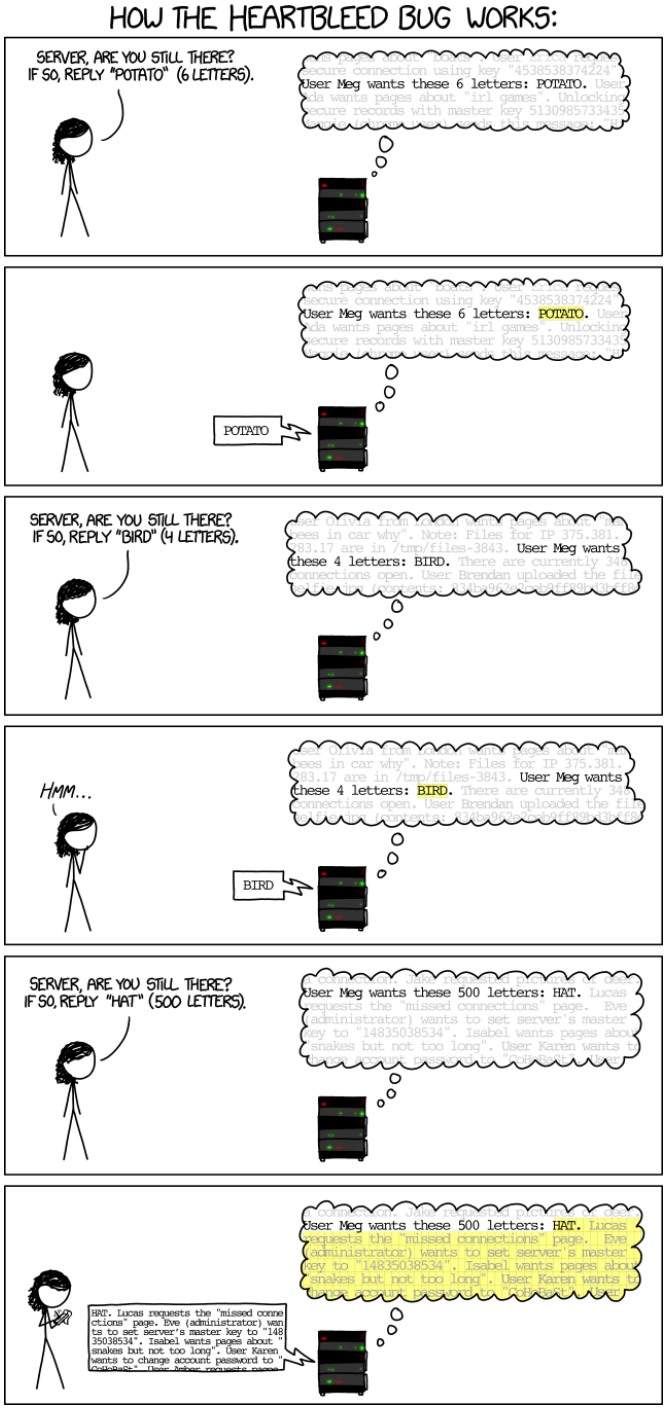


Bild: xkcd (Randall Munroe) – xkcd.com

Abb. 2.4: XKCD-Cartoon zur Funktionsweise des Heartbleed-Exploids

OpenSSL schickte immer eine Antwort mit der vom anderen Verbindungspartner angegebenen Länge – egal, ob dessen Anfrage tatsächlich so lang oder kürzer war. Zusätzlich zu den eigentlichen gefragten Daten, hat ein Server mit dem Heartbleed-Bug also im Zweifel noch mehr Daten aus seinem Speicher ausgegeben (bis zu 64 kByte), in dem zufällig Daten anderer Nutzer liegen können.

Dieser Angriff ist so genial einfach, dass ihn theoretisch jeder halbwegs begabte Möchtegern-Hacker hätte durchführen können. Und die Gerüchteküche brodelt schon: Die NSA habe diese Lücke absichtlich einbauen lassen und ausgenutzt – sie hat beides allerdings offiziell bestritten[2]. Auch sonst hat niemand zugegeben, bis zur offiziellen Veröffentlichung des Fehlers davon gewusst zu haben, aber natürlich ist diese Gefahr nicht von der Hand zu weisen.

Und die Ausbeute wäre prächtig: Nach der Veröffentlichung wurde schnell gezeigt, dass durch den Heartbleed-Angriff sehr einfach zahlreiche Benutzerdaten und Passwörter aus verschlüsselten Verbindungen des jeweiligen Servers abgegriffen werden können[3]. In Tests ist es nach einiger Zeit sogar gelungen den geheimen Schlüssel aus dem Zertifikat eines Servers zu stehlen[4], der die Grundlage für alle verschlüsselten Verbindungen zu diesem Server bildet. Mit Kenntnis dieses Schlüssels ist es ohne weiteres möglich, alle aufgezeichneten Verbindungsdaten zu entschlüsseln – die Verschlüsselung wäre (auch rückwirkend) vollkommen unwirksam.

Die technischen Details

(Dieser Abschnitt richtet sich an Programmier-Interessierte und kann auch übersprungen werden.)

Empfängt OpenSSL ein Heartbeat-Paket wird die Funktion `tls1_process_heartbeat()` (definiert in t1_lib.c) aufgerufen. Diese lässt sich mit

```
buffer = OPENSSL_malloc(1 + 2 ↙
   + payload + padding);
```

einen Speicherbereich zuweisen, dessen Größe der vermeintlichen (im Heartbeat-Paket angegebenen) Länge der übermittelten Daten (`payload`) plus 3 Byte für den Pakettyp und die Information über die Länge plus einem zufälligem Padding entspricht. Dann wird mit

```
memcpy(bp, pl, payload);
```

der Inhalt aus dem empfangenen Paket (darauf Verweist der Pointer `pl`) in den gerade reservierten Speicher (über den Pointer `bp`) kopiert. Und hier liegt das Problem: Genauer gesagt kopiert `memcpy` nämlich Daten mit der Länge von `payload` aus der aus dem empfangenen Paket in den neu zugwiesenen Speicher. Und ist `payload` größer als die tatsächliche Länge der Daten und liegen im folgenden Speicherbereich auch Daten von OpenSSL (was wegen der eigenen Speicherverwaltung des Programms sehr wahrscheinlich ist), werden diese Daten kurzerhand mitkopiert und beim anschließenden Senden der Antwort

```
r = ssl3_write_bytes(s, ↙
   TLS1_RT_HEARTBEAT, buffer, ↙
   3 + payload + padding);
```

auch mitgesendet. Zum Beheben des Problems haben die Entwickler nun in Version 1.0.1g die beiden Zeilen

```
if (1 + 2 + payload + 16 > ↙
   s->s3->rrec.length)
   return 0;
```

vor der eigentlichen Verarbeitung des Heartbeats eingefügt. Dadurch wird überprüft ob der angegebene `payload` (und die Länge der Kontrolldaten) größer ist, als die tatsächlich empfangene Datenmenge `rrec.length`. In diesem Fall bricht die Funktion ohne jegliche Rückmeldung ab.[5]

Die Folgen

Zwar wurde gleichzeitig mit der öffentlichen Bekanntmachung des Fehlers auch das Update mit dem Patch veröffentlicht, der den Fehler behebt; dennoch hat es mehrere Tage gedauert, bis alle wichtigen Website- und Server-Betreiber das Update auf ihren Servern installiert hatten (rapidshare.com z.B. war noch etwa zwei Tage lang verwundbar[3]).

Da der Heartbleed-Bug über 2 Jahre hinweg unentdeckt existierte, ist es sehr gut möglich, dass auch schon vorher jemand davon wusste und den Bug zum systematischen Sammeln von Passwörtern und Nutzerdaten (oder sogar Serverzertifikaten) verwendet hat. Da man dabei keinerlei Spuren hinterlässt, wird auch niemals geklärt werden können, ob das tatsächlich geschehen ist.

Das Fazit daraus: Alle Passwörter schnellst möglich ändern. Sobald der Betreiber der entsprechenden Website die neue OpenSSL-Version installiert hat und im besten Fall auch das SSL-Zertifikat erneuert hat (das ja auch geklaut worden sein könnte), sollte man sein Passwort dort ändern. Zur Sicherheit, für den Fall, dass irgendjemand es in den vergangenen zwei Jahren klauen konnte.

Oft wird gefragt, ob wirklich **alle** Passwörter geändert werden müssen. Antwort: Natürlich nicht. Zuerst einmal betrifft das nur alle Passwörter, die über eine SSL-Verbindung übertragen werden. Unverschlüsselt Übertragene Passwörter (Login-Seiten ohne https://) sind sowieso nicht geschützt, und lokale Passwörter wie etwa das Windows-Passwort zum Anmelden am Computer werden nicht übertragen und sind damit nicht betroffen.

Zudem sollte jeder selbst überlegen, wie viel an dem einen oder anderen Passwort hängt – Ob man nun jedem Account bei einem noch so

kleinen Forum im Internet ein neues Passwort verpassen sollte, sei mal dahingestellt. Wichtige Passwörter (z.B. vom Mail-Konto, Google- und Facebook-Account etc.) sollte man dagegen auf jeden Fall ändern. Etwas mehr zu diesem Thema beschreibt ein Heise Online-Artikel.[6]

Eine Liste vieler bekannter Websites und ihrer Empfehlungen bezüglich des Heartbleed-Bugs findet man bei mashable.com[7]. Ob eine Seite immer noch angreifbar ist, kann man mit verschiedenen Tools herausfinden, die in [8] aufgelistet sind.

[1] „Weg mit der Bruchbude!“ in c't 3/2014, Heise Verlag, heise.de/-2169173
[2] „NSA will nichts von "HeartbleedLücke gewusst haben“ 13.04.2014 im Heise Newsticker, heise.de/-2169173
[3] „Passwort-Zugriff: Heartbleed-Lücke mit katastrophalen Folgen“ 09.04.2014 auf Heise Security, heise.de/-2166861
[4] „Heartbleed-Worstcase: Server-Schlüssel kann ausgelesen werden“ 13.04.2014 im iX Newsticker, heise.de/-2169180
[5] „So funktioniert der Heartbleed-Exploit“ 10.04.2014 auf Heise Security, heise.de/-2168010
[6] „Passwörter in Gefahr - was nun?“ 10.04.2014 auf Heise Security, heise.de/-2167584
[7] „The Heartbleed Hit List: The Passwords You Need to Change Right Now”, mashable.com/2014/04/09/heartbleed-bug-websites-affected/
[8] „SSL-Gau: So testen Sie Programme und Online-Dienste“ 08.04.2014 auf Heise Security, heise.de/-2165995

Alle Internetseiten abgerufen am 21.04.2014

Leben

You know you're a biologist when...

von Danielle Cross

Life at universtity can be seen as an exotic ecosystem of sorts. Like-minded individuals meet and interact, form intraspecies relationships (ranging from symbiotic to parasitic), compete and assimilate. The students of a certain subject start to develop similar habits, mind-sets, and interests, all the while forming their own ecological niche and behavioral anomalies... Thus facilitating "species" identification. (Anyone who's jokingly tried to guess what random people study should know what I mean). The following shows sure-fire signs of my personal favorite student subspecies, and if you display any or all of the following symptoms, you just might be a Biology student.

You know you're a biologist when ...

... You start calling plants and animals by their scientific name. Constantly. Aren't those Bellis perennis and Papaver rhoeas beautiful?

... You can pronounce words like troglodytidae or glossopharyngeal and you know phrases like "Ontogeny recapitulates phylogeny". And you know what they mean.

... When you see fruit flies (D. melanogaster), you try to determine eye color and sex before shooing them away.

... You've shot pipette tips at your lab partner when no one was looking.

... Chemists have low expectations of you. You're "only" a biologist after all, and chemists tend to have God complexes. (My hypothesis being that they've inhaled too many dangerous chemicals over the course of their time in the lab. Said hypothesis is as yet unverified.)

... You harvest the organs of animals you dissect to decorate your apartment. You also bought prettier glasses to replace the jam jars they were in when you first brought them home.

... You use an Erlenmeyer flask as a vase.

... You get on non-biologists' nerves pointing out every different animal and plant you find when out and about.

... You refer to water as H_2O most of the time, because, for whatever reason, it seems to be faster.

... You start to question your doctor's decisions at times. Seriously, antibiotics against a virus? How exactly is that supposed to help?

... You can't take the creationist viewpoint seriously.

... You have considered naming a future pet Darwin or Mendel. Or Linné.

... You know way too much about how different species copulate. This knowledge is especially fun to share with non-biologists.

... You've embraced the fact that just about everything can be twisted into a sexual innuendo. This is fun at parties. Speaking of which ...

... You and your kind are known to party.

... Your lab preparation is perfect, but your results aren't usable.

... You do everything wrong in the lab, but your results are, strangely enough, actually fairly reasonable.

... You've talked about almost every bodily function and/or bodily fluid over lunch.

... You know that Harrison Ford had nothing to do with DNA structure. I repeat: Harrison Ford had absolutely nothing, nada, rien, nichts to do with DNA. He is an actor.

... You own a pot solely dedicated to boiling the flesh off of skulls, which was a tip a friend gave you over lunch. Said skulls are the also used for decorational purposes.

... You now only refer to cups as 'Tassulas'.

... You've swiped pH indicator strips from the lab and tested pH levels of bodily fluids, beverages and diverse other liquids for leisure.

... Everything is elementary and significant. Everything.

... There is such a thing as dumb questions.

... You're fairly certain that, at some point, you will refer to your future children as the F1. And that you'll fool around with Mendelian crossing schemes at some point.

... You've stolen something small from the lab. Like pH indicator strips. Or safety goggles. Or the odd test tube.

... You probably don't like botany. You passed the test, but your plants always die.

... You are slightly pyromaniacal, so it was really disappointing for you when nothing exploded during Chemistry. But you did get hydrogen chloride on your hand at one point and had to wash your hands for fifteen minutes, which was kind of funny and kind of scary at the

same time, so ...

... You have a vague idea of what chloroform smells like.

... You own a stuffed animal shaped like a cell. Or maybe two.

... You've watched the number of your fellow biology students shrink at least by half over time.

... You know what is meant by "Campbell", "Purves", and "Schmeil-Fitschen", and probably own at least one of them.

... When asked what you're doing later, you answer "Science".

... You have to leave your flowing ball dresses at home. Tragic, really.

... Your lab coat is full of stains and holes of unknown origin. And that one you made on purpose. Admit it.

... The work load of non-scientific subjects seems to be almost cute in comparison.

... You cringe when TV shows try to sound impressive by explaining genetic processes but didn't do their homework.

... You know the difference between fruit and vegetables. And you eat lots of both. Actually, chances are you're vegetarian to some extent.

... You've used your dissection instruments around the house.

... Gloves are optional.

... You own a magnifying glass, an Opinel, and/or binoculars.

... You know how to use a microscope and have used one to look at random things when you were bored in class, which made you realize how ugly fingers look up close.

... You have a favorite source of caffeine. If it's not coffee, it's coke, if it's not coke, it's tea. Show me one biologist who says they don't need caffeine and I'll show you a liar. Or a hippie. Or a hippie liar.

... The concept of wearing homemade sandals out of old tires and packaging cord never occurred to you. You're not a geo-ecologist, after all. But you do enjoy watching their morning yoga sessions in front of the lecture hall.

... Ideally, you know why humans are more closely related to echinoderms than to annelids. And what chlorophyll is.

... Technically, witchcraft, voodoo, the Force and aliens aren't legitimate answers. But in your mind, they explain a heck of a lot of things. Especially when you don't feel like studying.

... You've realized that everyone is a Homo. It's the sapiens part you have to work on.

... Your equipment has been too big for practical use at some point or other. (Which brings us back to sexual innuendos, if you know what I mean).

... You can go to the zoo for free. Which is awesome. Really, really awesome.

... You call blondes, red-heads and people with blue eyes mutants. But lovingly, of course.

... You either have or know someone who has a science-based tattoo. I know people with the structural formula of vanillin or that of hemoglobin drawn onto their skin, as well as one tribal strand of DNA.

... You use a scalpel to cut the tags off of clothes.

Kreativ

Eine zweistimmige Invention

von Florian Kranhold

Es ist mittlerweile schon wieder eine gute Weile her, dass ich das letzte Mal etwas komponiert habe. In letzter Zeit habe ich wieder mehr kompositorische Ideen, aber am sichersten fühle ich mich nach wie vor im zweistimmigen Kontrapunkt für Klavier.

Wie man diesen am besten setzt, bewundere ich nach wie vor in Bachs *Wohltemperierten Klavier*. So hat mir zum Beispiel das Präludium in G-Dur im zweiten Buch einige Spielfreude bereitet und ich mochte das kurze Motiv mit repetitivem Wechselton.

Ich habe also versucht, eine Invention mit ähnlicher Motivgestaltung zu schreiben und hoffe, dass mir das einigermaßen gelungen ist. Dabei habe ich versucht, ein paar ausgefallenere Modulationen hineinzubasteln, sodass man kurzzeitig in c-Moll, wenig später aber bereits in a-Moll angekommen ist. An anderen Stellen wollte ich die beiden Stimmen abwechselnd „miteinander sprechen“ lassen, sodass es da manchmal etwas leer klingen mag, dafür aber hoffentlich etwas klarer und dialogisierter.

Die ersten Takte sind hier abgedruckt; eine vollständige Version gibt es auf fkranhold.de.

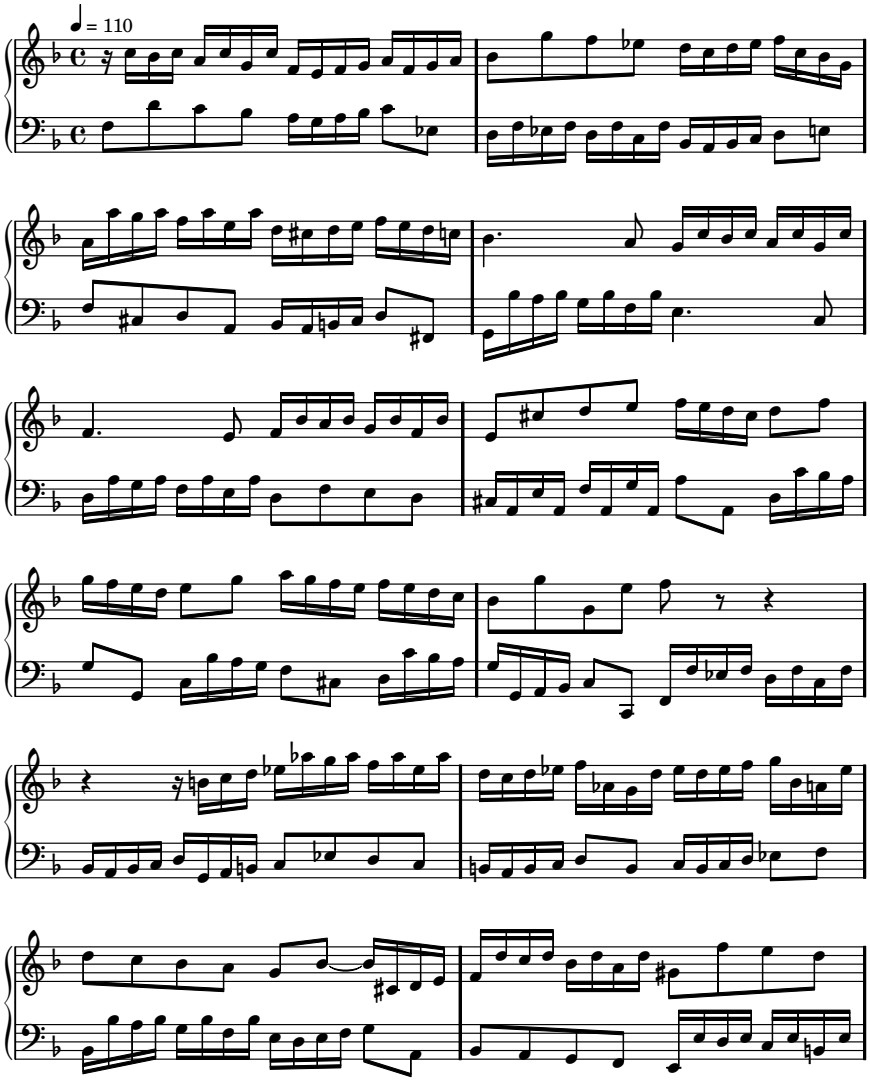


Abb. 4.1: Die ersten Takte meiner Invention in F-Dur

Informatiker und Kuchen

Faires Teilen ohne Endstück

von Jannik Buhr

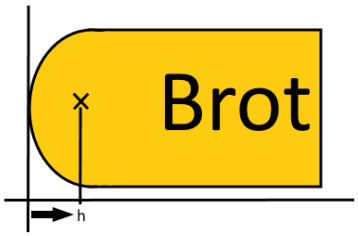
Manch einer bezeichnet Mathematiker und Informatiker zuweilen als leicht verrückt oder gar lebensfremd. Dass dem aber nicht so ist, möchte ich an zwei sehr praxisnahen Beispielen erläutern:

Wie jeder andere Mensch auch ist der Informatiker gerne Kuchen. Hierbei stellt sich aber zumeist (wenn man nicht gerade alleine ist) die Frage, wie man einen solchen fair aufteilen kann. Ohne Waage oder Lineal wollen wir es erreichen, dass ein jeder sich fair behandelt fühlt. Zunächst zur Erklärung des

Begriffes „fair“: Man nehme den Menschen als rein egoistisch handelndes Wesen, so sei für ihn eine Aufteilung fair, wenn er mehr oder genau so viel bekommt, wie er meint, dass ihm zusteht. Genau wie Jesus einst am See Genezareth das Brot teilte wird es uns möglich sein, einen Kuchen so aufzuteilen, dass alle beteiligten Informatiker das Gefühl haben, sogar mehr bekommen zu haben, als ihnen zusteht. Für eine Zahl von n Informatikern setze jeder einen Zahnstocher mit seinem Namen an die Stelle das Kuchen der Länge l, an dem er meint, dass die Länge $l \cdot n^{-1}$ erreicht sei.

Das Stück wird nun an der Stelle abgeschnitten, die sich genau zwischen dem am weitesten links liegenden Fähnchen (1) und dem nächsten liegenden (3) befindet. Dieses Stück geht nun an die Person (1), die ihr Stück am kleinsten geschätzt hatte. Auf diese Weise bekommt sie sogar mehr oder gleich viel, als sie dachte, dass ihr zusteht! Die übrigen $n-1$ Informatiker fahren induktiv fort, bis nur noch einer übrig ist, der dann das letzte Stück bekommt.

Damit wäre eines der beiden großen Probleme unserer Zeit gelöst; ein Weiteres ist die Problematik des Endstückes. Wie es scheint, möchte niemand das Endstück eines Brotes essen, aber auch hierfür hat der Mathematiker eine Lösung!



Sei $h > 0$. Für ein an der yz-Ebene anliegendes Endstück (siehe xy-Querschnitt in der Abbildung) wird nun ein Schnitt an der Ebene $\{(x,y,z) \in \mathbb{R}^3 \mid x = h\}$ gemacht. h ist hierbei also der Abstand vom linken Rand des Endstücks zur Schnittfläche. Legen wir nun einen Schnitt an, so ist alles rechts dieses Schnittes kein Endstück, links davon allerdings schon. Es gilt aber:

$$\lim_{h \to 0} (\text{Brot links}) = 0$$

Das Stück links des Schnittes wird dann nämlich so klein, dass es nicht mehr als Endstück, sondern nur noch als Krümel zu bezeichnen ist, bzw. vollkommen gegen Null läuft und verschwindet. Problem gelöst.

Kant gibt sich die Kante

von Lukas Heimann

Ein Haus, Winter Siebzehnvierundachtzig,
eine Nacht, die grad' heranbricht
  über dem alten deutschen Land,
  in dem sich Kant derzeit befand.

Winternächte brachten zu diesen Zeiten
noch reichlich Unannehmlichkeiten.
  Vor allem diese Kälte, denn
  ohne Heizung plagte sie nebst Kindern
    dann auch noch die Älteren.

So saß auch Kant in seiner Küche,
in der ihm wohlbekannten Nische,
  und fragte sich, auf welche Weise
  man kalte Suppe am besten verspeise;

Denn das Feuer war schon längst erloschen,
weil er heute seinen letzten Groschen
  ausgegeben hatte für
  weißes DIN-A4-Papier.

Holz war also viel zu teuer,
blieb nur noch das alte Feuer-
  wasser, das er krasserweise
  still und leise
jahrelang in seinem Haus versteckte,
damit es in der Kälte seine Lebensgeister weckte.
  So tat er, was er wohl schon lange plante:
  Philosoph und Denker *Kant gibt sich die Kante.*

Er steigt die Stiege in den Keller,
erst zögerlich, dann immer schneller,
  und stürzt durch die geheime Pforte
  zu seines Alkoholes Horte.

Mit Bedacht wählt er die erste Flasche,
zieht den Korkendreher aus der Tasche,
  dreht den Korkenzieher in den Korken,
  und er hält ein altes Wort, denn
er hatte einem Freund versprochen –
länger her – mehrere Wochen –
  diesem mal – in allen Ehren –
  den Sinn des Lebens zu erklären

Was eine Schande:
Er hängt an der Bande
  zum Tresen
  verfasst seine Thesen.

Geistesblitze blitzen auf

Gedankengänge nehmen ihren Lauf auf
kurz darauf
schaut Kant auf;
Ihn trifft der Schlag
und er sagt:

„Habe Mut“
– ja, das ist gut –
„Dich Deines eigenen Verstandes“
– und er verstand es –
„zu bedienen“
– nur wie denn?!

Na klar, wer Kant kennt und las,
der weiß: „In vino veritas!“

Ein kleines Rätsel

Auflösung zur letzten Ausgabe

von Marcel Hörz

In der letzten Ausgabe haben wir unseren Lesern ein kleines Rätsel gestellt und versprochen, es in dieser aufzulösen. Diese lautet:

[1] rot
[2] lila
[3] lila
[4] rot
[5] grün
[6] rot
[7] rot
[8] weiß
[9] grün

Impressum

Chefredakteur:
Florian Kranhold

Layout:
Tobias Gerber, Florian Kranhold
Erstellt mit LaTeX

Logo:
Michael Thies

Autoren:
Florian Kranhold, Lukas Heimann, Michael Thies, Danielle Cross, Jannik Buhr, Marcel Hörz

Gastautoren:
Arjun Sarin

Redaktionsanschrift:
Florian Kranhold
Rottenburger Straße 8
72070 Tübingen

Kontakt:
www.neologismus-magazin.de
www.facebook.com/neologismus.magazin
info@neologismus-magazin.de

Veröffentlicht am 3. Mai 2014