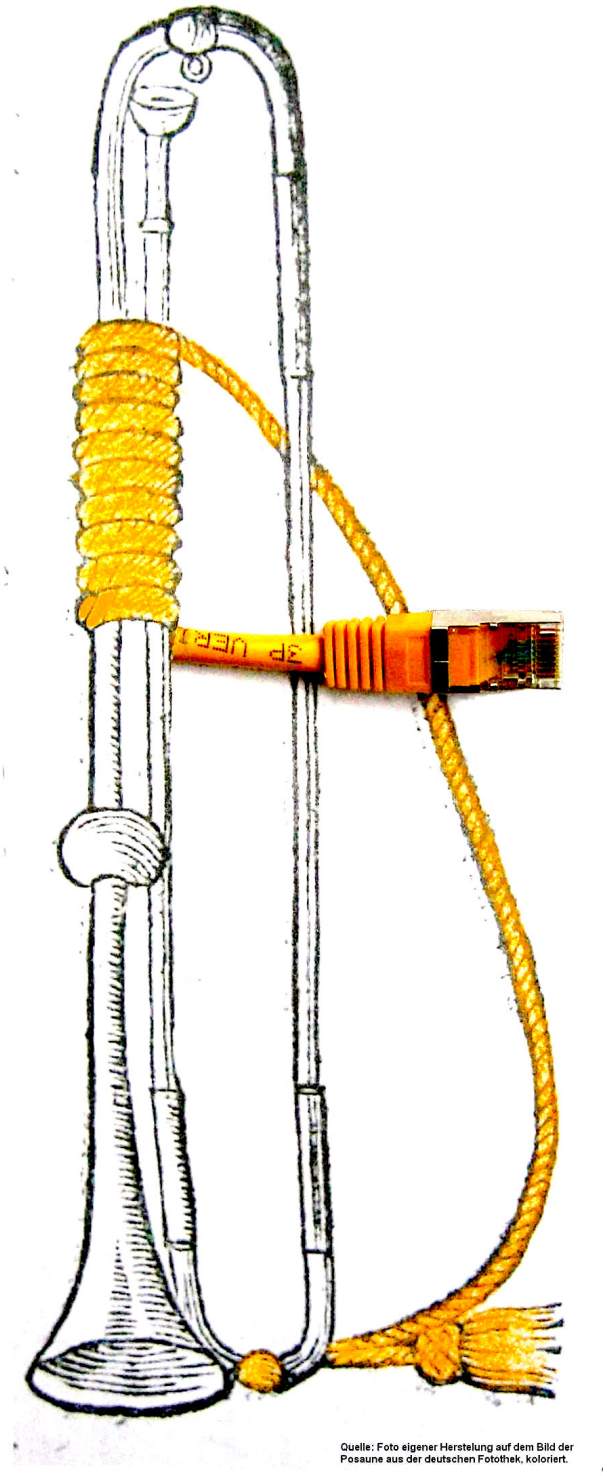
Vierte Dimension

3/2013

Das Forth–Magazin

für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten

In dieser Ausgabe:



Quelle: Foto eigener Herstelung auf dem Bild der Posaune aus der deutschen Fotothek, koloriert.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D–22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH–Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmesstechnik und bauen z. Z. eine eigene Produktpalette auf.

Know–How Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential–Dosierwaagen, DMS–Messverstärker, 68000 und 68HC11 Prozessoren, Sigma–Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX–Verleih

Seit unserem Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX–Komponenten, dass ich meine privat eingebrachten Dinge nun Anderen, vorzugsweise Mitgliedern der Forth–Gesellschaft e. V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO–RCX–Set, so wie es für ca. 230,–€ im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO Steine.

Anfragen bitte an
Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist „narrensicher“!

RetroForth
Linux · Windows · Native
Generic · L4Ka::Pistachio · Dex4u
Public Domain
`http://www.retroforth.org`
`http://retro.tunes.org`

Diese Anzeige wird gesponsort von:
EDV–Beratung Schmiedl, Am Bräuweiher 4, 93499 Zandt

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690–380
Fax: 02461/690–387 oder –100
Karl–Heinz–Beckurts–Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug– und Sondermaschinen, Fuzzy Logic.

FORTecH Software GmbH
Entwicklungsbüro Dr.–Ing. Egmont Woitzel

Bergstraße 10	D–18057 Rostock
Tel.: +49 381 496800-0	Fax: +49 381 496800-29

PC–basierte Forth–Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software–/Systementwurf. Mehr als 15 Jahre Erfahrung.

Hier könnte Ihre Anzeige stehen!

Wenn Sie ein Förderer der Forth–Gesellschaft e.V. sind oder werden möchten, sprechen Sie mit dem Forth–Büro über die Konditionen einer festen Anzeige.

`Secretary@forth-ev.de`

Ingenieurbüro
Klaus Kohl–Schöpe

Tel.: **(0 82 66)–36 09 862**
Prof.–Hamp–Str. 5
D–87745 Eppishausen

FORTH–Software (volksFORTH, KKFORTH und viele PDVersionen). FORTH–Hardware (z.B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs– und Messtechnik.

Impressum

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth–Gesellschaft e. V.
Postfach 32 01 24
68273 Mannheim
Tel: ++49(0)6239 9201-85, Fax: -86
E–Mail: `Secretary@forth-ev.de`
`Direktorium@forth-ev.de`
Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208
IBAN: DE60 2001 0020 0563 2112 08
BIC: PBNKDEFF

Redaktion & Layout

Bernd Paysan, Ulrich Hoffmann
E–Mail: `4d@forth-ev.de`

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluss

Januar, April, Juli, Oktober jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00€ + Porto u. Verpackung

Liebe Leser,

1969, ARPANET, ein Projekt des US–Verteidigungsministeriums. Vernetzt wurden Universitäten und Forschungseinrichtungen, von Anfang an keinesfalls anonym. Jedenfalls für den überschaubaren Kreis der damals Beteiligten.

1990 beschloss die US–amerikanische National Science Foundation, das Internet für kommerzielle Zwecke nutzbar zu machen, wodurch es über die Universitäten hinaus öffentlich zugänglich wurde. Tim Berners–Lee entwickelte um das Jahr 1989 am CERN die Grundlagen des World Wide Web. Dann kamen die Browser, zum kostenlosen Download bereitgestellt. Heute baut man Betriebssysteme darum herum, FireFox OS kommt diesen Monat auf's Handy.

2000 ist die offene Kommunikation und Selbstdarstellung von Personen und Firmen fest etabliert. In den folgenden Jahren werden die Server dieser Welt zugeschmissen mit kommerziellen Webseiten, Spam, Blogs, email, Twitter, Telephonie, und was noch so alles. Abhören der Telekommunikation wird fast schon überflüssig, beinahe alles ist schon veröffentlicht, bereitwillig angeliefert, und so gewollt von uns allen. Die *wireclamp* hat ausgedient. Weil alles massenweise herausposaunt wird, besteht das Problem darin, Relevantes überhaupt noch zu finden. Auf der anderen Seite liegen auch intime Daten massenweise digital vor. Mein Krankenakte ist gewiss nicht öffentlich, und meine Webseiten-Statistik auch nicht, jedenfalls solange wie ich das nicht selber poste.

2013 geht praktisch nichts mehr ohne Internet, es gibt eigentlich nur noch digitale Kanäle der Kommunikation — vom Papier mal abgesehen — und es gibt viele gute und legitime Gründe, eine wirklich vertrauliche Übermittlung von Informationen zwischen Partnern zu haben. Und diese auch selbst herstellen zu können, wann immer man es will. Eine Art gehobene Alphabetisierung. Bernd hat sich dieser Frage verschrieben und zeigt einen Weg.

Auf der letzten Forthtagung haben wir beschlossen, auch englischsprachige Artikel zu bringen. Gut für mich, muss ich doch nicht mehr übersetzen. Und der Autor kann es selbst noch lesen und vorzeigen. Andrew macht den Anfang, er lebt in Australien, und unterrichtet auch mit Hilfe von Forth in MCUs. Ulli lädt ein zu mathematischen Spielereien, eine doppelte Herausforderung, muss man doch verstehen, bevor man es programmieren kann. Rafael holt ganz Praktisches aus seinem Erfahrungsschatz, und lässt uns teilhaben an der Handhabung von MCUs, bindet das wiederum ein in Forth. Und mit Hannes können wir weiterarbeiten an der Tonerzeugung, und das Ton–Projekt voranbringen, wenn wir uns einbringen mögen, denn es ist ja seit dem letzten Heft alles publik, was er da so macht.

2014 dann — *und posaunt das laut heraus, wo ihr könnt!* — wird die Forth–Gesellschaft 30 Jahre alt. Wer hätte das gedacht?[1] Ein Grund zu feiern, und das wollen wir dann in Wien auch machen. Ich freu mich schon auf Euch!

Euer Michael

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können sie auch von der Web–Seite des Vereins herunterladen.
`http://fossil.forth-ev.de/vd-2013-03`

Die Forth–Gesellschaft e. V. wird durch ihr Direktorium vertreten:

Ulrich Hoffmann Kontakt: `Direktorium@Forth-ev.de`
Bernd Paysan
Ewald Rieger

[1] Die Satzung wurde am 28. Oktober 1984 von 22 Personen unterzeichnet.

SWAP und seine Freunde

Ich hätt' da mal eine Auflösung für das Rätsel (aus Heft 2013/01):

Der gesuchte Drache ist am Münchner Rathaus, ich habe ein Beweisfoto gemacht, und mit passendem Geo-Tag versehen:

`https://plus.google.com/photos/114020517704693241828/albums/5897612733468653329/5897612734410504802`

Da dem Drachen der zweite Kopf fehlt, ist er streng genommen kein Swap-Drache.

bp

Bernd, danke für Deine Auflösung. Ja, das ist richtig. Aufmerksame Besucher des Marienplatzes konnten sich sicherlich erinnern. SWAP hängt wohl nicht nur mit Swap-Drachen ab...

Der Auflösung unseres ersten SWAP-Freund-Rätsels folgt im gleichen Flügelschlag die etwas kniffligere Suche nach dem nächsten SWAP-Freund:

Frage wiederum: **Wo befindet sich dieser SWAP-Freund?**

Antworten bitte an die Redaktion unter vd@forth-ev.de.

Die Antwortgeber der richtigen Antworten werden im kommenden Forth-Magazin veröffentlicht.

Ihr habt andere Freunde von SWAP erblickt, die es zu lokalisieren gilt? Schickt uns ein Foto und einen kleinen Text.

uho

Proper fixation — a substitute for anaesthesia

> Dieser blog hat einen Namen, weil man das so macht, und dazu kam es folgendermaßen. Eines Tages erzählte ich einem Bekannten, dessen Eltern beide Ärzte waren, etwas darüber wie ich ein technisches Problem gelöst hatte. Er fand meine Lösung nicht gerade gut, und meinte dazu: „Ordentliche Fixierung als ein Ersatz für Anästhesie..." Hallo? Aber der Spruch gefiel mir. Passte in mein Weltbild. ...

In den *tech blog* von Yossi Kreinin geriet ich beim Surfen zum Thema *Forth & stack machines*. Yossi hat, wie er schreibt, den Weg beschritten, Forth-Prozessoren selbst zu erzeugen, für echte Anwendungen (real world), und sieht das nun kritisch. In einem ausführlichen blog-Beitrag legt er dar, wie es dazu gekommen ist. Leseprobe:

> But I actually had a vested interest in stacks, and I began to like registers more and more. The thing is, expression *trees* map perfectly to stacks: `(a+b)*(c-d)` becomes `a b + c d - *`. Expression *graphs*, however, start to get messy: `(a+b)*a` becomes `a dup b + *`, and this `dup` cluttering things up is a moderate example. And an *expression graph* simply means that you use something more than once. How come this clutters up my code? This is reuse. A kind of factoring, if you like. Isn't factoring good?...

Möge das Schnipsel anregen, sich den kompletten Beitrag durchzulesen. mk

Quelle: Yossi Kreinin, My history with Forth & stack machines, September 10th, 2010
`http://www.yosefk.com/blog/about`
`http://www.yosefk.com/blog/my-history-with-forth-stack-machines.html`

Neues aus der Zisterne

In der VD 4/2012 hatte ich darüber berichtet, dass ich ein atmega32-Board mit einem Ultraschall–Entfernungsmesser ausgerüstet hatte und ich damit regelmäßig den Wasserstand aus der Dunkelheit gefunkt bekomme. Das ging ganz ordentlich bis in den Winter. Da wurden die Daten (Entfernung) zuerst zeitweise schlecht und dann dauerhaft. Eingefroren? Verrostet? Von den Spinnen zugewoben? Kabel von den Ratten angenagt?

Allerdings verzögerten Schnee und Eis die Begutachtung bis ins späte Frühjahr. Ans Tageslicht gezerrt, stellte sich heraus, dass Wasser in das Sensorgehäuse eingedrungen war und zusammen mit saurer Hydrolyse der Platine des Entfernungsmessers den Garaus gemacht hatte. So schad. **Abb. 1, 2**

Neulich orderte ich Ersatz und dieser Tage habe ich den Sensor ausgetauscht. Die Messung und die Funkübertragung funktionierten sofort wieder. Nur den Akku musste ich noch laden. Das Kästchen für den Sensor habe ich nun trickreich abgedichtet, den Sensor ebenfalls eingeklebt, so dass von außen kein Wasser mehr eindringen sollte — wenn es nicht schon drin ist. Jetzt ist alles wieder gut. Und dank Martins Erinnerung an die guten alten Hebelgesetze dauert es jetzt auch nur noch 15 Minuten,

bis die Deckel gelupft und die Leiter hinunter ins nasse Dunkel gestellt ist.

Erich Wälde

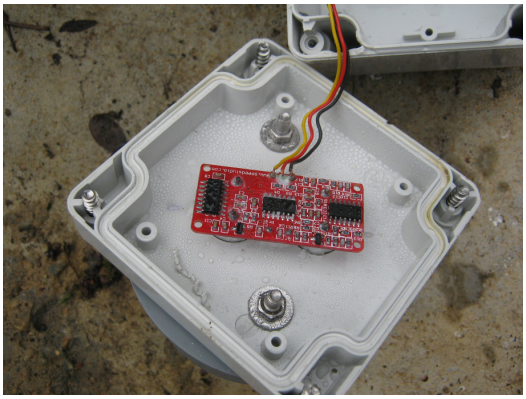


Abbildung 1: Das Bild habe ich direkt nach dem Öffnen vom Käschtle gemacht!

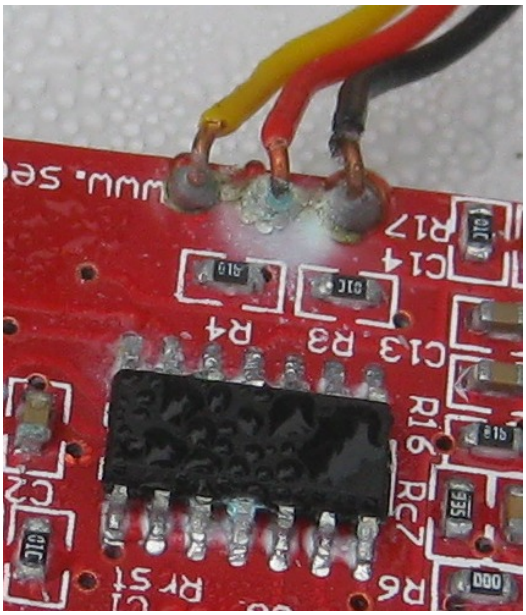


Abbildung 2: Im Zoom sieht man die Wassertropfen und Korrosionsstellen.

Die Forth–Gesellschaft auf LinuxTag&MakerFaire

Bernd Paysan

Die Forth–Gesellschaft braucht Nachwuchs, und dafür muss man an die Öffentlichkeit. Das machen wir jedes Jahr am LinuxTag. Dieses Jahr ist die MakerFaire als Event dazugekommen, Heise hat das Konzept von O'Reilly nach Deutschland gebracht. Der Triceps wurde natürlich mal wieder umgebaut, und als Programm–Herausforderung kam das interaktive Go–Spiel dazu.

LinuxTag

Friedel Amend hat die etwas windigen Schnüre durch Alu–Stäbe ersetzt, die mit Magneten festgehalten werden. Auch der Greifer ist ausgefeilter, nicht nur eine flache Scheibe mit Magnet, sondern ein zweiteiliges Ding, das unten die Kugel halbrund umgreift, und oben an den Stäben hängt. Damit man schön mit Ruck die Kugel absetzen kann, ist der obere Teil absenkbar — fährt man dann schnell nach oben, wird der untere Teil in einem Ruck mitgerissen, und die Kugel bleibt sauber liegen. Diese mechanischen Verbesserungen sind für die Herausforderung Go–Spiel auch nötig.

Die wesentlichen Ideen hatte Friedel in der Woche vor dem LinuxTag, die neue Hardware bekam ich also erst auf dem LinuxTag selbst zu sehen. Also haben wir unser Präsentationskonzept etwas verändert: Es gibt jetzt zwar auch den Triceps zu sehen, aber das Interessantere ist der Bernd, der dem Triceps das versprochene Go–Spiel beibringt. Das lief dann in mehreren Phasen.

- Zuerst musste die Ansteuerung des Roboters im b16 so abgeändert werden, dass sie mit den Stangen und dem neuen Greifer zurechtkommt.
- Dann musste die Spielfelderkennung im Tablet an die Lichtbedingungen in der Halle angepasst werden.
- Die Spielsteine liegen beim Go–Spiel auf einer Pyramiden–Ablage (so, wie man Kanonenkugeln stapelt), das muss wieder der b16 machen.
- Und zu guter Letzt muss das Tablet mit dem FPGA kommunizieren.

Dabei haben mir mehrere Leute stundenlang beim Programmieren zugeschaut, was bestimmt dem eigentlichen Zweck, nämlich Leute für Forth zu begeistern, gedient hat. Am Samstag war es dann auch so weit: Der Triceps spielte tatsächlich Go auf dem 9x9–Brett (siehe Abbildung 1). Im Wesentlichen gegen die Standbesatzung, aber das macht ja nichts.

Auf dem billigen Android–Tablet aus China läuft Gforth. Das nimmt das Kamera–Previewbild, und verzerrt es mit OpenGL perspektivisch. Damit ist es wieder ein rechteckiges Raster. Dieses Pixelbild wird in drei Bins zerlegt: Weiß, Schwarz und Grün. Und von diesen Bins wird dann eine Mehrheitsentscheidung gefällt, was es denn jetzt nun ist. Das Ergebnis, also die Auswertung des Spielfelds, sieht man rechts oben am Tablet. Das Tablet sendet dann über die serielle Schnittstelle Binärcode an den b16, den

der ausführt — und der Triceps macht dann die Spielzüge. Den Quelltext sollte ich ausführlich erläutern, aber dafür ist in diesem Heft kein Platz, deshalb nur der Verweis auf's Repository [1].

Das neue Standkonzept ist für mich recht anstrengend (in lauter Umgebung programmieren, während einem immer einer über die Schulter blickt), aber zumindest gefühlt ein Erfolg. Für den LinuxTag selbst gilt, was ich auch schon letztes Jahr gesagt habe: Der Messe–Teil verliert immer weiter an Bedeutung. Damit sich überhaupt noch Besucher des nach wie vor beliebten Konferenzteils zur Ausstellung hin verirren, haben die Organisatoren die Stände in die Halle direkt neben den Vortragsräumen im Keller verlegt.

Als Spiele–Engine habe ich das Go–Spiel von Ian Osgood verwendet [2]; das nutzt zwar den Monte–Carlo–Algorithmus, ist aber im Vergleich zu z.B. GNU Go nicht sehr spielstark.

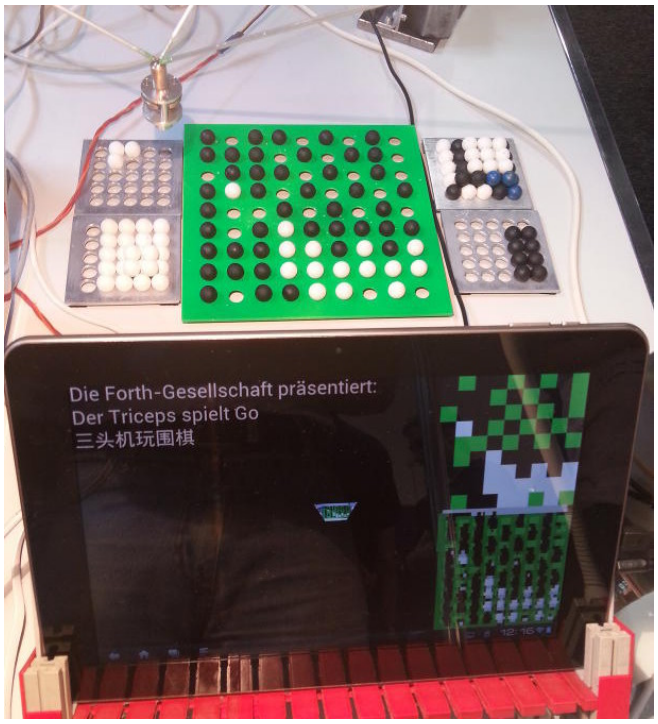


Abbildung 1: Der Triceps (weiß) verliert ein Go–Spiel

MakerFaire

Da die Hardware nicht ganz perfekt funktioniert hat, hat Friedel für die MakerFaire die Servos ausgetauscht, die Aluminiumstangen durch Kohlefaserrohre ersetzt, und den Greifer leichter und präziser gemacht. Außerdem hat er die weißen Kugeln mit einem Lack besprüht, der nicht klebt. Die MakerFaire war allerdings sehr voll, und hatte so viel Publikum, dass das ein langes Go–Spiel kaum ganz verfolgen hätte können. Also spielte der Triceps wieder Solitaire. Die alten Servos hatten eine lästige Hysterese, und die neuen Servos neigen zum Schwingen. Beide verzittern deshalb immer wieder Spielsteine. Die Farbauswertung muss noch mühsam an die Lichtverhältnisse angepasst werden, und das Tablet genau ausgerichtet werden, sonst klappt es nicht.

Die Besucher der MakerFaire sind etwas anders als die beim LinuxTag, es finden sich mehr Bastler (die natürliche Zielgruppe für Forth), aber auch mehr junge Leute, auch weiblichen Geschlechts (siehe Abbildung 2). Während auf dem LinuxTag die anderen Stände unser Konzept immer mehr nachmachen, einen beweglichen Publikumsmagnet aufzustellen (zumeist ein 3D–Drucker), ist das bei der MakerFaire von vornherein geplantes Standkonzept der Konkurrenz. Der Triceps muss sich also mit 3D–Druckern, Quadcoptern und vielem anderen mehr messen.

Da die MakerFaire Anfang August stattfand, war es ausgesprochen heiß. Trotzdem war es auch sehr voll, und von morgens bis kurz vor Schluss ständig etwas los. Wir haben an einem Tag mehr Anmeldeformulare verteilt als an vier Tagen LinuxTag. Die Bastler auf der MakerFaire hätten gern den Triceps nachgebaut, was natürlich bedeutet, dass sich Friedel überlegen muss, wie man diverse gedrehte und gestanzte Teile durch welche ersetzt, die ohne teures Werkzeug gefertigt werden können. Und für den Heimgebrauch sollte das Go–Spiel auch auf einem 19x19–Brett stattfinden… dafür kann man aber keine Kugeln verwenden. Der Plan geht jetzt in Richtung Stifte, die entsprechend eng gepackt werden können.

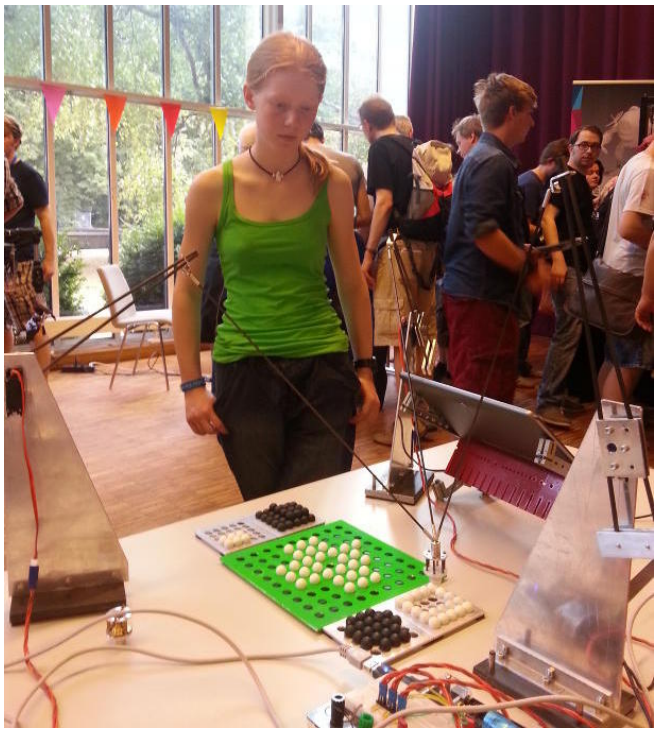


Abbildung 2: Junge Zuschauerin auf der MakerFaire

Literaturverzeichnis

[1] Bernd Paysan, *Quellcode des Android–Programms zum Go spielen,* `https://fossil.net2o.de/minos2/artifact/75df84f3fe5bc98186092d515dd6d4ff3962331e`

[2] Ian Osgood, *Ian Osgood's Forth Go Program (FGP),* `http://www.quirkster.com/iano/forth/fgp.html`

Weiß, Schwarz, Braun

Ulrich Hoffmann

Der Computer ist ein prima Hilfsmittel, um Knobelaufgaben zu lösen. Dabei lässt sich herausfinden, wie gut sich welches Werkzeug für welche Aufgabenstellung eignet. Wir wollen eine klassische Knobelei ansehen und überlegen, wie das in Forth zu lösen ist.

Martin Gardner (1914–2010) ist für seine Kolumne *Mathematical Games* im Scientific–America–Magazin bekannt. Bei uns erschienen sie als *Mathematische Spielereien* im Spektrum der Wissenschaft.

In seinem Buch *Sphere Packing, Lewis Carroll, and Reversi* [1] findet sich folgendes Rätsel (Übersetzung des Autors):

> **Weiß, Schwarz, Braun**
>
> Prof. Toni Weiß vom Institut für Mathematik, Prof. Kim Schwarz, Philosophie, und Uli Braun vom Universitäts–Sekretariat saßen beim Mittagessen zusammen. „Ist es nicht bemerkenswert“, sagte die Dame, „dass unsere Nachnamen *Schwarz*, *Weiß* und *Braun* sind und dass außerdem auch noch einer von uns schwarzes, einer braunes und einer weißes Haar hat?“
>
> „In der Tat — “, antwortete die Person mit dem schwarzen Haar, „und ist Dir auch aufgefallen, dass keiner von uns die Haarfarbe hat, die zu seinem Namen passt?“
>
> „Menschenskind, Du hast recht!“, rief Prof. Weiß aus.
>
> Wenn nun das Haar der Dame nicht braun ist, welche Haarfarbe hat dann Prof. Schwarz?

An dieser Stelle sollte man eine kurze Pause beim Lesen einlegen und erstmal in Ruhe über das Rätsel nachdenken.

—

Wir wollen versuchen, dieses Rätsel mit einem Programm in Standard–Forth zu lösen.

Zunächst einmal geht es im Rätsel um Haarfarben und um Namen, die jeweils weiß (Weiß), schwarz (Schwarz) oder braun (Braun) sein können. Dann geht es darum, wer von den Dreien männlich und wer weiblich ist, denn der Text lässt es offen, welche Person welches Geschlecht hat[1]. Die Vornamen der drei Personen — Toni, Kim und Uli (im Original Merle, Leslie und Jean) — sind allesamt sog. Unisex-Vornamen, können also für Männer wie auch für Frauen stehen. Der Text spricht lediglich von *der Dame*, so dass wir annehmen dürfen, eine Dame ist im Gespräch mit zwei Herren.

Welche Situationen kommen überhaupt in Frage? Nun, grundsätzlich könnte jede Person weiße, schwarze oder braune Haare haben und jede der drei Personen könnte die Dame sein. Unser Situations-Raum ist also die Menge der 4-Tupel (w, s, b, g) mit $w, s, b, g \in \{\text{weiß}, \text{schwarz}, \text{braun}\}$, wobei w angibt, welche Haarfarbe Prof. Weiß hat, s, welche Haarfarbe Prof. Schwarz und b, welche Haarfarbe Uli Braun hat. g (Geschlecht) gibt an, wer von den Dreien die Dame ist.

Die drei Farben weiß, schwarz und braun können wir durch die Zahlen 0, 1, und 2 darstellen und wir definieren entsprechende Konstanten und Ausgabewörter für Haarfarben und Namen (siehe Listing Zeilen 5–17).

Wie kann nun eine Situation repräsentiert werden? Wir könnten die vier Teile (w, s, b, g) einer Situation als einzelne Stack–Elemente darstellen. Die Erfahrung zeigt, dass sich Forth–Wörter mit mehr als drei zu bearbeitenden Stack–Elementen schnell mit vielen Stack–Operatoren anfüllen und dadurch unübersichtlich werden[2]. Wir wollen die Situationen daher als ein einziges Stack–Element darstellen, in dem w, s, b und g kodiert sind. Wir wählen dazu einfach eine Zahl zur Basis 3, deren Ziffern dann *wsbg* sind.

Die Zahl 42, im 3er-System 1120_3, entspricht dann der Situation, dass Prof. Weiß weiblich ist (letzte Ziffer ist 0). Sie und Prof. Schwarz sind beide schwarzhaarig (die ersten beiden Ziffern sind 1) und Uli Braun hat braune Haare (dritte Ziffer ist 2).

Insgesamt gibt es $3 \cdot 3 \cdot 3 \cdot 3 = 81$ Situationen. Wir definieren das Forth–Wort `gbsw` (Zeile 33–36), das eine solche Situation nimmt und die Werte der vier Teile zur weiteren Verarbeitung extrahiert und (in umgekehrter Reihenfolge) auf den Stack legt.

Die obige Situation $42 = 1120_3$ ist möglich, aber nicht zulässig: Zunächst mal darf Prof. Weiß nicht die gleiche Haarfarbe wie Prof. Schwarz haben, da ja jede Haarfarbe nur einmal vorkommen soll. Außerdem dürfen weder Prof. Schwarz schwarze, noch Uli Braun braune Haare haben, denn ihre Haarfarben dürfen ja nicht zu ihren Namen passen. Es gibt also unzulässige Situationen aber — wenn unser Rätsel Lösungen hat — auch zulässige Situationen. Welches genau die zulässigen Situationen sind, sehen wir uns als Nächstes an.

Über die Aussagen, die die drei Personen machen — wir nehmen an, sie sagen die Wahrheit — erfahren wir von

[1] Die im Buch *Mathematische Knobeleien* [2] zu findende Übersetzung beginnt mit dem Satz *Professor Weiß vom Mathematikdepartment, Philosophieprofessor Schwarz und Frau Braun vom Sekretariat saßen zusammen beim Mittagessen.* was im Gegensatz zum Originaltext nahelegt, Braun wäre eine Frau. Damit wäre das Rätsel unlösbar. Nicht jeder Braun ist eine Eva.

[2] Ein Ausweg sind benannte lokale Variablen etwa mit `locals|` oder `{:`, die aber wiederum das Faktorisieren von Worten erschweren.

folgenden Bedingungen, die in zulässigen Situationen erfüllt sein müssen:

1. Eine Person hat schwarzes, eine weißes, eine braunes Haar, d. h.: Jede der drei Haarfarben ist einmalig.
2. Jede Person hat eine Haarfarbe, die sich von ihrem Namen unterscheidet.

Durch die Gesprächsfolge — wir unterstellen übliches Konversationsverhalten — und den erklärenden Text erfahren wir zudem:

3. Es gibt eine Dame (*die Dame*) in der Mittagsrunde.
4. Die Dame hat kein schwarzes Haar, denn die Person mit schwarzem Haar antwortet auf den ersten Satz der Dame.
5. Die Dame ist nicht Prof. Toni Weiß, weil Prof. Weiß ebenfalls erstaunt auf den ersten Satz reagiert, die Dame diesen zweiten Satz also nicht auch spricht.
6. Die Person mit schwarzem Haar ist ebenfalls nicht Prof. Toni Weiß, denn sie würde ja nicht auf ihren eigenen (zweiten) Satz mit Erstaunen reagieren (wir unterstellen ja übliches Konversationsverhalten).
7. Die Dame hat kein braunes Haar, denn es heißt in der abschließenden Frage: „Wenn nun das Haar der Dame nicht braun ist…“.

Bedingung 3. ist für unsere Situationen immer gegeben. Per Konstruktion können keine Situationen ohne oder mit mehr als einer Dame auftreten.

Aus den restlichen Bedingungen formulieren wir jetzt Forth-Worte, die zu einer gegebenen Situation ermitteln, ob in dieser Situation die gegebene Bedingung vorliegt (siehe Zeilen 56–78).

Eine zulässige Situation, also eine Lösung, ist eine Situation, in der alle Bedingungen erfüllt sind. Für eine einzelne Situation prüft das das Wort `lösung?` (Zeilen 83–89)

Nun gilt es, alle 81 Situationen zu überprüfen, was im Wort `wsb` (Zeilen 99–103) geschieht. Hier zeigt sich auch der Vorteil, Situationen durch ein einziges Stack–Element darzustellen: Keine verschachtelten `DO-LOOP`s, die die einzelnen Situations–Teile durchnummerieren. Eine Schleife, die die Situationen in der oben beschriebenen Darstellung durchläuft, reicht vollkommen.

Und welche Haarfarbe hat nun also Prof. Schwarz? Gibt es überhaupt eine Lösung oder gar mehrere? — Das wird nicht verraten. Einfach das Programm laufen lassen, selber knobeln oder aber eine bessere Lösung finden und hier vorstellen.

Literatur

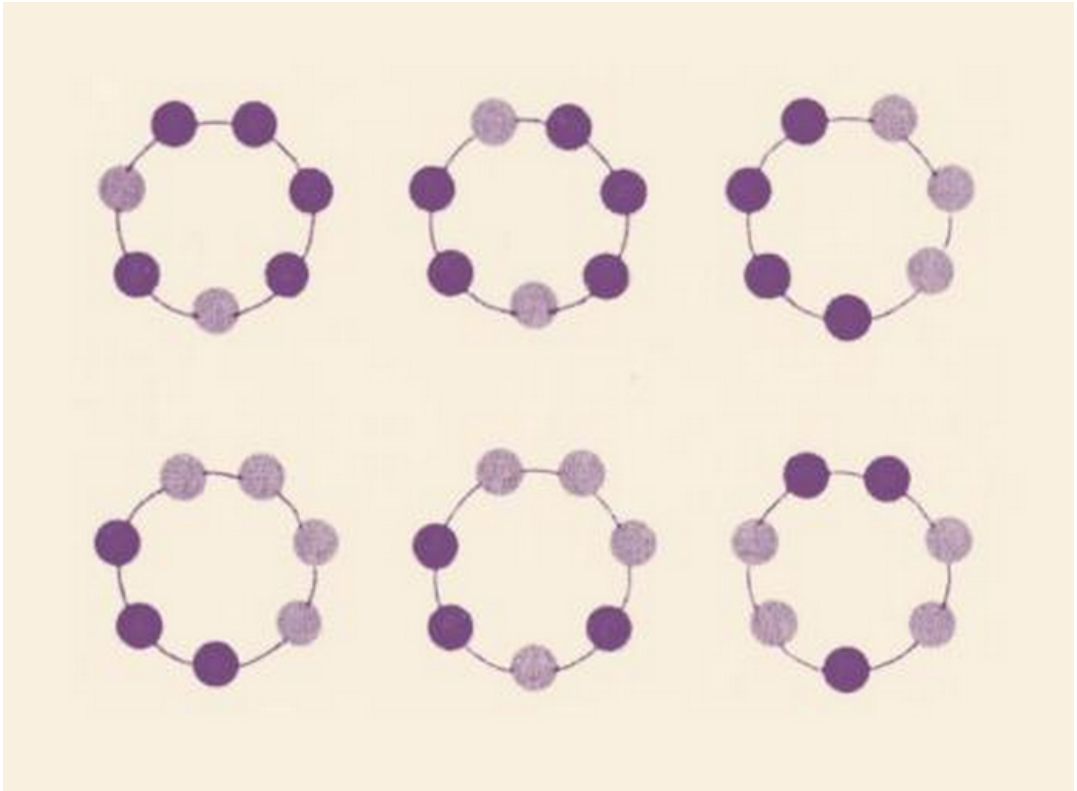
[1] *Sphere Packing, Lewis Carroll, and Reversi*, Martin Gardner, Cambridge University Press, 2009
[2] *Mathematische Knobleien*, Martin Gardner, Friedr. Vieweg + Sohn GmbH, Verlag, Braunschweig, 1973

Listing

```
1  \ Lösung für das Weiss-Schwarz-Braun-Rätsel von Martin Gardner  uh 2013-07-23
2  \ Public Domain
3  \ encoding: UTF8
4
5  0 Constant weiß
6  1 Constant schwarz
7  2 Constant braun
8
9  : .haarfarbe ( n -- )
10       dup 0 = IF drop ." weißes Haar" EXIT THEN
11       dup 1 = IF drop ." schwarzes Haar" EXIT THEN
12       drop ." braunes Haar" ;
13
14 : .name ( n -- )
15       dup 0 = IF drop ." Prof. Toni Weiß" EXIT THEN
16       dup 1 = IF drop ." Prof. Kim Schwarz" EXIT THEN
17       drop ." Uli Braun" ;
18
19 : .anrede ( f -- )
20       IF ." Frau "  ELSE ." Herr " THEN ;
21
```

```
\ Die Darstellung von Situationen

\                     /----------- Wer ist die Dame (0: Prof. Weiß,
\                     |            1: Prof. Schwarz, 2: Uli Braun)
\                     | /--------- Haarfarbe von Uli Braun
\                     | |
\                     | | /------- Haarfarbe von Prof. Schwarz
\                     | | |
\                     | | | /----- Haarfarbe von Prof. Weiss
\                     | | | |
\                     V V V V
: gbsw ( situation -- g b s w )
     3 /mod ( g situation1 )
     3 /mod ( g b situation2 )
     3 /mod ( g b s w ) ;

: haarfarbe ( situation person -- haarfarbe ) \ Welche Haarfarbe hat eine Person?
   >r  gbsw  r> pick  >r  2drop 2drop r> ;

: dame ( situation -- person ) \ Wer ist weiblich?
   gbsw 2drop drop ;

: dame-haarfarbe ( situation -- haarfarbe ) \ Welche Haarfarbe hat die Dame?
   dup dame  haarfarbe ;

: .situation ( situation -- )
    3 0 DO ( situation )
       cr ."     "
       I over dame = .anrede
       I .name ."  hat "  dup
       I haarfarbe .haarfarbe ." ."
    LOOP ( situation )
    drop ;

\ Die Bedingungen

: haarfarben-einmalig? ( situation -- f ) >r \ 1. Jede der drei Haarfarben ist einmalig.
   r@ weiß haarfarbe  r@ schwarz haarfarbe  <>
   r@ weiß haarfarbe  r@ braun haarfarbe  <> and
   r@ schwarz haarfarbe r> braun haarfarbe  <> and ;

: haarfarben-unpassend? ( situation -- f ) >r \ 2. Jede Person hat eine Haarfarbe, die sich
   r@ weiß haarfarbe weiß <>                  \    von ihrem Namen unterscheidet.
   r@ schwarz haarfarbe schwarz <> and
   r> braun haarfarbe braun <> and ;

: dame-haarfarbe-nicht-schwarz? ( situation -- f ) \ 4. Die Dame hat kein schwarzes Haar
   dame-haarfarbe schwarz <> ;

: dame-nicht-prof-weiß? ( situation -- f ) \ 5. Die Dame ist nicht Professor Toni Weiß
   dame weiß <> ;

: prof-weiß-haarfarbe-nicht-schwarz? ( situation -- f )
    \ 6. Hat Professor Toni Weiß schwarze Haare?
   weiß haarfarbe schwarz <> ;

: dame-haarfarbe-nicht-braun? ( situation -- f ) \ 7. Die Dame hat kein braunes Haar
   dame-haarfarbe braun <> ;
```

```
\ Test einer Situation auf Zulässigkeit

: lösung? ( situation -- f ) >r
   r@ haarfarben-einmalig?
   r@ haarfarben-unpassend? and
   r@ dame-haarfarbe-nicht-schwarz? and
   r@ dame-nicht-prof-weiß? and
   r@ prof-weiß-haarfarbe-nicht-schwarz? and
   r> dame-haarfarbe-nicht-braun? and ;

\ Das Rätsel, probiere alle 81 Situationen

3    ( Haarfarben für Prof. Weiß )
3 *  ( Haarfarben für Prof. Schwarz )
3 *  ( Haarfarben für Uli Braun )
3 *  ( Möglichkeiten, wer die Dame ist )
Constant #situationen

: wsb ( -- )
    #situationen 0
    DO
      I lösung? IF  cr cr ." Lösung:" I .situation THEN
    LOOP ;

cr .( Rätsel lösen: wsb ) cr
```



4E4th for the Texas Instruments MSP430 LaunchPad Educational Environment

Andrew Reid

The 4E4th small micro-computer based Texas Instruments LaunchPad populated with the powerful 16 bit, 16 kByte MSP430G2533 Flash Device offers many advantages to Teachers and Students in Physics, Chemistry, Biology and Environmental Science who are interested in complimenting the theoretical aspects of their course with straight forward easy repeatable practical experiments.

Hardware and Software makeup

- PC computer. Teacher or Student supplied hardware required is any standard Windows XT or 7 PC computer with the normal Standard Operating System. Note The 4E4th Educational Environment has no special requirements and your Operating System is not changed in any way by this hardware or software.
- PC Application Software used. Any existing Text Editor or Word Processor with a simple text output (.txt) such as NotePad.
- Hardware needed. Texas Instruments LaunchPad populated with the 16 Bit, 16k Byte MSP430G2553 Flash Device along with all the standard cables and connection items. These items are all delivered as standard with the Texas Instruments LaunchPad . Typical cost between 5*to*15 — Availability Worldwide.
- Texas Instruments Software. Texas Instruments Free LaunchPad Windows Driver.
- The 4E4th software which is also free will be listed towards the end of this document with download details .

A View of the 4E4th Educational Environment

We will illustrate the 4E4th experimental environment and show measurement flexibility when using the 4E4th- IDE "Integrated Development Environment" and the 4E4th "GraphWriter" along with using the free Onewire.4th Application Software to measuring Temperature using the low cost well known Maxim Integrated DS 18B20+ sensor.

4E4th flexibility is enhanced as it supports as standard communication with all of Maxim Integrated one wire devices. We will assume that with the focus on global warming students may take a mild interest in their personal Thermal Comfort, have an interest in Global Warming and may have concerns about their Car's Environment on a hot day and perhaps how effective their air-conditioning is and the electrical energy it consumes to meet their satisfaction. With this in mind we will measure and plot using the 4E4th Graph–Writer two temperatures one the Dry Bulb the other the Wet Bulb, and output the data in standard CSV (Comma Separated Value) format which may be used by most Spreadsheet programs such as Excel and then use this numerical data to calculate and plot from the spread sheet %Relative Humidity (%RH) and Dew point temperature. Both parameters again viewed by many as important in GreenHouse Environmental Control.

The 4E4th Data Logging language has been Flashed so that it resides permanently on the LaunchPad. This means that the launchPad may use previously written programs such as the free Onewire.4th supplied by Imaging Associates and available for Download from a number of Websites. These programs are made up of "Command Words" which may be used and changed in real time by teachers and Students. The Basic program in this case Onewire.4th once downloaded remains stored on the LaunchPad unless it is deliberately erased. This means that a Class Set of LaunchPads say 15 could all have a different Experiments — all Tested and ready to run.

The 4E4th-IDE LaunchPad control window **fig. 4** displays all the 4E4th Command Words which would not normally be used by the Student in a well defined experiment.

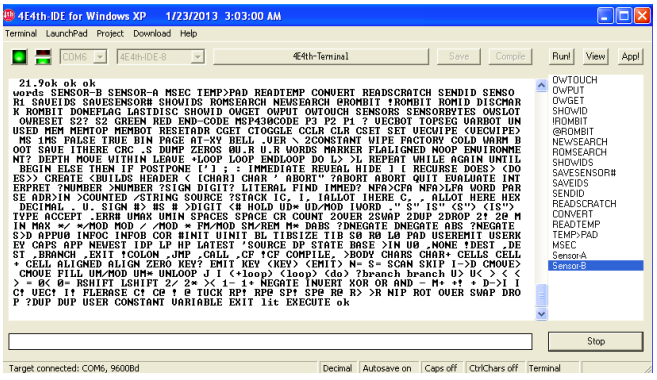


Figure 1: 4E4th-IDE LaunchPad control window

Students doing "Standard School Experiments" would in the main be using temperature measurement commanded directly and on demand from the right hand column commands of which are based upon the standard free Onewire.4th program command "RUN", see **fig. 5**.

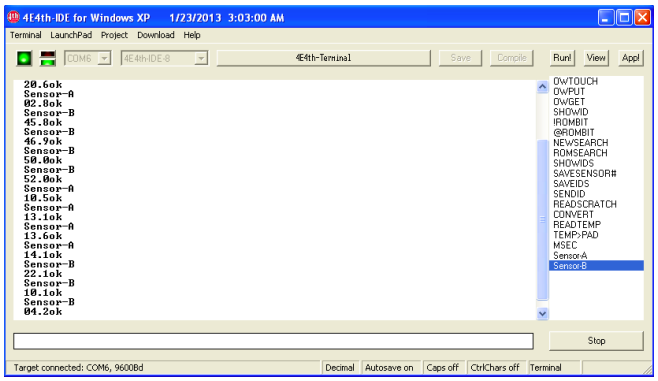


Figure 2: Temperature measurement commanded directly

In this situation the student may manually initiate Sensor A or Sensor B temperature readings — useful in setting up the experiment and performing adjustments to apparatus. In this mode the student may also gain an appreciation of the DS18B20+ sensor and perhaps reflect upon the sensor response time and the accuracy of the measurements. On that point an appreciation of the standard DS18B20+ temperature matching and tracking should be appreciated as we will be using the temperature difference between the Wet and Dry Bulb (TdB -TwB) measurements in our calculations. The GraphWriter picture of two onewire DS18B20+ temperature sensors joined together with blue Tack (**fig. 1**)may give confidence in the measurements to be.

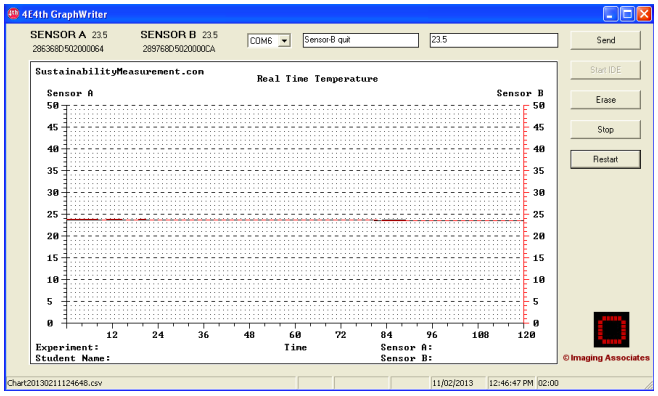


Figure 3: Temperature measurement commanded directly

It should be noted that the two onewire DS18B20+ sensors are connected to the same input pin on the LaunchPad. They may be connected by inexpensive domestic telephone wire with each having a different position with wire lengths determined by the demands of the experiments in the class rooms to a maximum of 5 meters. Outside measurements with battery portable computers are easily to do experiments or individual project investigations such as considering some aspects of the solar impact on Automobile and Greenhouse environment where for instance Relative Humidity, Soil Temperature Dew Point etc may be of interest.

The Heat transfer aspects of the domestic refrigerator along with it's demands for energy and the manufactures claims for their CO2 impact on the planet Earth. Physics, it's fundamental relevance to the real world.

Back to our Relative Humidity and Dew Point Temperature experiment

The temperature sensors must be organized so that we measure a Dry Bulb and a Wet Bulb. the Wet Bulb should be under a slight draft. A low cost USB powered portable computer fan has proved effective in the past. A simple construction is shown —- most students would do better (**fig. 3**).

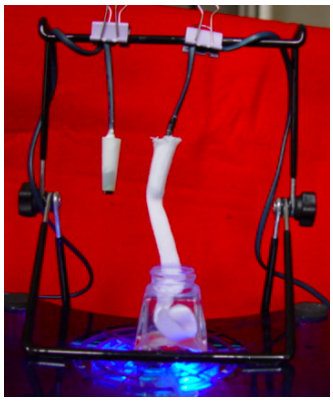


Figure 4: The fan has a blue light and the Wet Bulb wick is Cotton.
The 4E4th GraphWriter will give us Trace information with readings in this case about every two seconds.

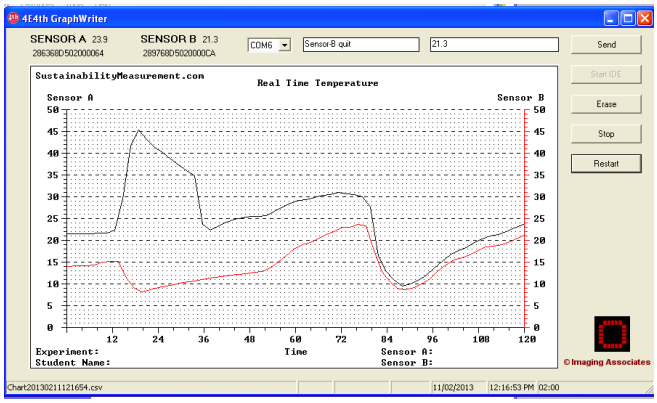


Figure 5: Estimating DS18B20+ response time.

The Graphwriter Picture **fig. 2** was chosen to give a feel for the DS18B20+ response time.

4E4th GraphWriter has also given us the numerical information, **fig. 6**.

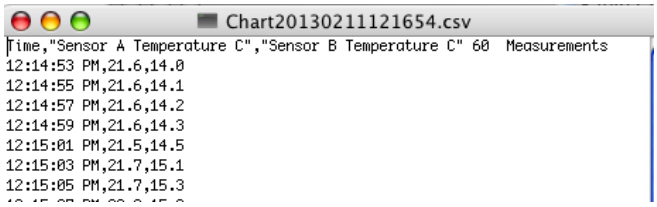


Figure 6: GraphWriter numerical information.

Students may manually or spreadsheet calculate and plot the Relative Humidity (RH) and Dew Point Temperature (Tdp) as follows. The background to; and the derivation of these equations may have formed part of the Outcomes an earlier Lesson Plan.

1. The DS18B20 measures temperature in Centigrade. Tdb is the 4E4th LaunchPad measured Dry Bulb Temperature. Twb is the 4E4th LaunchPad measured Wet Bulb Temperature. These Temperatures are measured using 4E4th GraphWriter with numerical data over time available in CSV format.
2. The Atmospheric Pressure is assumed to be P(atm) = 101.3 kPa (kiloPascals) but the locally measured value should be used if available .
3. A conversion factor A is used and is calculated as follows: A = 0.00066 (1.0 + 0.00115 Twb)
4. The Saturation Vapour Pressure SVPwb is calculated at the measured Wet Bulb temperature Twb: SVPwb = e[(16.78 Twb - 116.9)/(Twb + 237.3)]
5. The Actual Vapour Pressure is calculated: AVP = SVPwb - A x P(atm) x (Tdb - Twb)
6. The Saturation Vapour Pressure SVPdb is calculated at the Dry Bulb temperature Tdb: SVPdb = e[(16.78 Tdb - 116.9)/(Tdb + 237.3)]
7. The relative humidity is then calculated as a percentage: RHwhere AVP is the actual vapour pressure and SVPdb is the Saturated Vapour Pressure at Tdb.
8. Dew point temperature Tdp is then calculated: Tdp = (243.5 x Ln(AVP/6.112)) / (17.67 - Ln(AVP/6.112)) where AVP is the actual vapour pressure and Ln is the natural log.

A Fixed Format dedicated Imaging Associates International program not having the 4E4th educational flexibility shows Relative Humidity & Dew point temperature plots using these equations over a 24 hr period, **fig. 7**.

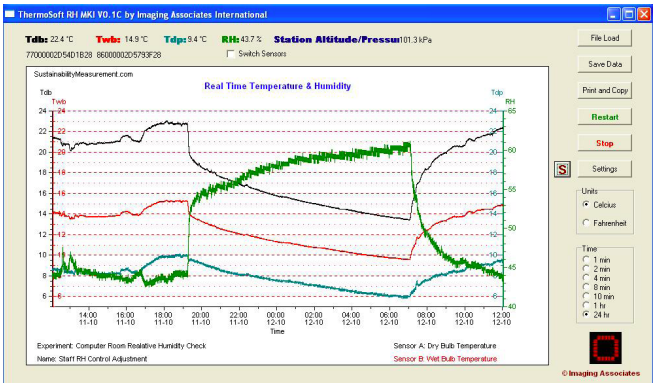


Figure 7: Relative Humidity and Dew point temperature plots.

Training

Training if required may be arranged at the school with a maximum of two teachers per 4E4th Two temperature Experimental Measurement Station. Where schools have Laboratory support staff they will be welcome.

The training is normally for one day and may be extended. The content has a fixed format for the first part of the day with the second part addressing the previously agreed particular needs of teaching staff which must address their required Learning Outcomes of their subjects of interests through the straight forward application of 4E4th launchPad experiments.

Experiments usually address the fields of Physics, Chemistry and Biology with an overall focus where these fields of study impact upon Environmental and Sustainability issues. In this context the traditional practical measurements and experiments often associated with these subjects can be complimented with many simple experiments which may stimulate the student having an on going interest in the subjects as they find their impact fundamental to societies well being. Further ideas and relating to specific requirements may be explored by teachers with the author.

Support

In order to appreciate and gain insight in to the support which may required by a particular school: always after a some staff have had straight forward training, some insight into the nature the 4E4th Texas Instruments LaunchPad Two Temperature Environment will be useful.

- Imaging Associates International prefers to always supply the 4E4th Two Temperature System with the following Hardware and Software components:
- Texas Instruments LaunchPad with MSP430G2553 which has been Flashed by Imaging Associates with the 4E4th Command Language along with the Two Temperature OneWire.4th application two temperature measurement program.
- Two Maxim DS18B20+ Temperature Sensors and 1k0 component configuration prewired to connectors which plug directly into the LaunchPad. This is the minimum configuration with sensors not configured for student use but ideal for teachers to obtain Imaging Associates International support whilst they make a decision to purchase a number for Student use.
- 4E4th-IDE (IDE stands for “Integrated Development Environment”) which is the “Command Word” communicator between Windows and the 4E4th LaunchPad with the attached DS18B20+ temperature Sensors which was described earlier in this document.
- GraphWriter giving Real Time Display.
- 4E4th, 4E4th-IDE , GraphWriter, onewire.4th software along with the Texas Instruments USB Launchpad Driver will be supplied on a USB Flash Drive.
- Standard 4E4th “ Command Word Language “ and the 4E4th-IDE Integrated development Environment PDF documentation will also be supplied.
- As the Imaging Associates 4E4th LaunchPad system is ready to go after it is plugged into a USB port

and the Texas Instruments Driver is installed. Imaging Associated International will provide a Two Page Document which will enable :

- Two temperatures to be read using the 4E4th-IDE.
- Two Temperatures to be plotted using GraphWriter.

Some Aspects of the 4E4th Data Logging Environment which may relate to its flexibility for Student independent Investigations.

Consider the following; Imaging Associates International is using the temperature sensor DS18B20+ in what is called its "Parasitic Power Mode". This means that only two wires of standard low cost telephone wire may be used — in our case the red and green wires. To get a temperature measurement the sensor is connected similar to that show in **fig. 8**.

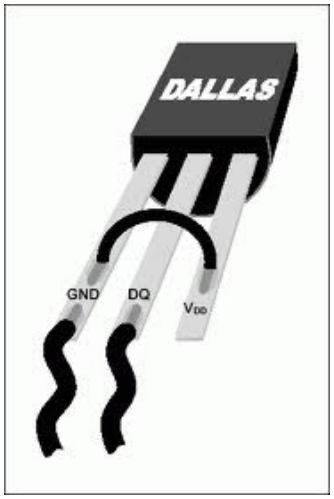


Figure 8: Connecting the DS18B20+ sensor.

The Important point — if the two wires are accidentally connected the wrong way no damage befalls the DS18B20+.

Temperature measurements usually require a wide range of differently shaped sensors to assist in obtaining accurate and repeatable reading even in our range of -10 to 110 degrees centigrade.

The DS18B20+ is easy to package using commonly available materials. The following pictures show some sensors constructed in this manner with the exception of the Stainless Steel Spike most were bonded with strong glue and water proofed with 710 Silicone Glass Sealant good for a temperature range of -50 to 150 degrees centigrade using items such as Aluminum Foil, Copper Strip, Ping Pong balls , irrigation Fittings, and Sealant Nozzles. [1]

Summary

The 4E4th Educational OneWire Two Temperature System may be attractive for Student Open Investigation Work in Science and may be focused under teacher guidance, to address a number of Learning Outcomes associated with the Heat and Temperature aspects of the curriculum along with how these types of measurements may be used to address the broader issues of sustainability where environmental issues are of concern. These type of measurement address a wide section of the student community as the fundamental measurements made, may be presented to the very young in an educational palatable manner but with out loss of substance. For the advance student the teacher has the flexibility of dealing with real data to different mathematical levels using Calculus , and Curve fitting say in the context of Newton's Law of Cooling or a practical evaluation of whether "A Big Mac" comes in a better insulated package than a "Hungry Jack".

Figure 9: 4E4th Environment Logo.

The 4E4th Measurement Environment offers:

- 4E4th & Maxim Onewire.
- The 4E4th-IDE with immediate command word RUN facility.
- GraphWriter with numerical spreadsheet compatible output.
- OneWire.4th Command Program to collect data within the temperature is in the range -10 to 110 degrees Centigrade.
- ======= Most are Free ======

This environment was created by Brad Rodriguez CamelForth-onewire/Canada, Michael Klaus 4E4th/Germany, Dirk Bruehl 4E4h-IDE&GraphWriter/USA
Email or call Andrew Reid for Download Websites .

Appendix: Protecting DS18B20+

Fig. 10 showes what has been archived by students in this project. Older Students using a screw driver, Craft Cutting Knife, Domestic Glue, Silicon Sealant and sometime a simple Soldering Iron should manage. School support staff will no doubt be able to offer improvements.

[1] See Pictures Appendix.

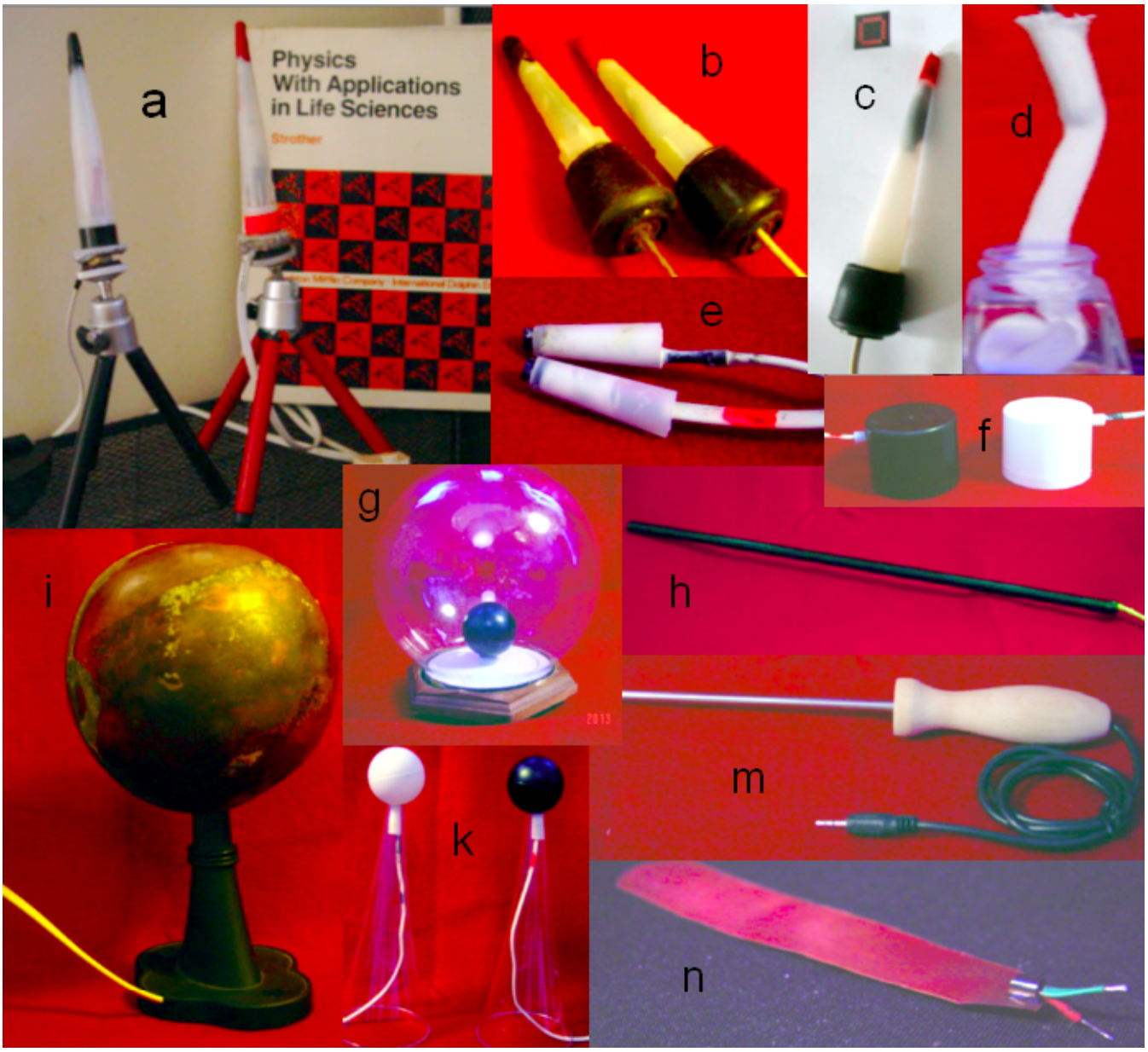
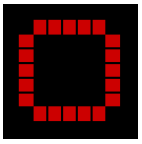


Figure 10: Synopsis of what has been archived by students in this project. **a)** Non metal Spikes. **b)** Paint Nozzle Spike — Rubber foot, Tip seal Liquid Electrical Tape, additional water proofing Silicon Glass Sealant. Connection Wire domestic Telephone Cable. Comfortable in young peoples hands, non-toxic Robust. **c)** Non metal Spike — water proof DS18B20+ tip protected internal. **d)** One approach to Wet Bulb. Wick Cotton. **e)** Small Water proof fast response Ds18B20+ tip external. **f)** Plastic Spray Paint Caps with well defined Emissivity. **g)** Planet Earth with single ping pong Globe Thermometer. **h)** Irrigation Rod Based Water proof tip. **i)** Unique Globe using Brass system float. **k)** Classic Ping Pong Globe using plastic nozzle tip and champagne flute support. **m)** Metal Spike Stainless Steel – imported for information. **n)** Copper Leaf.

Links



Contact andrewreid@imagingassociates.com.au
http://www.sustainabilitymeasurement.com
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

net2o — das Internet neu erfinden, Teil 2: Verschlüsselung

Bernd Paysan

Im 2. Teil der Artikelreihe geht es darum, wie net2o die Daten verschlüsselt. Das Thema gewinnt durch die Leaks von Edward Snowden *erheblich an Aktualität, wissen wir doch jetzt, dass überall mitgelauscht wird, nicht nur in China oder dem Iran. Dort aber dann halt doppelt.*

Warum verschlüsseln?

Man hätte es wissen können: am 21. Dezember 2002 schon habe ich in meinem Sendung–mit–dem–Bush–Blog geschrieben

> *Dazu gibt's jetzt die Heimatsicherheit. Das ist so was wie die Stasi, nur viel größer und für Amerika. Die horcht jetzt jeden ab, und weiß alles. Macht aber nichts, weil sie auch nicht weiß, was sie alles weiß.*

Gut, es ist nicht die Heimatsicherheit selbst, die das Internet totalüberwacht. Es ist auch nicht nur ein Dienst, sondern es sind eine ganze Reihe von Diensten, die jeweils bei sich den großen Datenstaubsauger aufstellen, und abschnorcheln, was geht, und dann die anderen Dienste mit den Daten füttern, die sie „legal" erheben dürfen. Oder was sie dem entsprechenden Kontrollgremium halt verschweigen.

Man darf sich da auch nicht in Sicherheit wiegen: Wer am FKK–Strand nichts zu verbergen hat, aber vom Nachbarn zu sehr angestarrt wird, kann ein paar Meter weiterrücken. Wenn die NSA dein Twitter–Posting falsch versteht, oder die Bestellungen bei Amazon sowohl einen Rucksack als auch einen Druckkochtopf enthalten, hast du ein Sonderkommando im Haus. Oder du darfst nicht in die USA einreisen, obwohl du gar keinen Spaten dabei hast, um Marilyn Monroe auszugraben (wie auf Twitter angekündigt). Du hast es mit Leuten zu tun, die den Turing–Test nicht bestehen, nicht nur mit Computern. Ja, die bestehen den Turing–Test natürlich auch nicht, aber die NSA vertraut ihnen, dass sie die Terroristen finden. Und die NSA hat schon CEOs von Großfirmen, die nicht mitspielen wollten[1] in den Knast gebracht. Und die Firma wurde von einem Konkurrenten geschluckt. Staatsfeinde vernichtet man eben. Ob der Betroffene wirklich ein Staatsfeind ist, entscheidet ein Geheimgericht ohne Anhörung des Betroffenen, und natürlich muss ein Geheimdienst auch Erfolgsmeldungen liefern... und ein Geheimdienst, der ein Rechenzentrum mit 5 Zetabyte Kapazität plant, der will einfach alles speichern.

Auch beschränken sich die USA bei Staatsfeinden nicht einfach nur auf bärtige Bombenleger. Whistleblower sind genauso Staatsfeind Nr. 1, und wer Whistleblower unterstützt, ist ebenfalls ein Staatsfeind. Und weil jeder, der um drei Ecken einen Terroristen kennt, auch verdächtig ist, sind wir, die Forth–Gesellschaft, wohl auch eine terroristische Vereinigung, ist die FG doch von dem Mann gegründet worden, der als 2. Vorsitzender der Wau–Holland–Stiftung nicht nur Spenden an Wikileaks weitergeleitet hat, sondern auch öfter mal öffentlich aufgetreten ist. Und wer weiß, ob es nicht ausreicht, besagtes lustiges Blog über den Herrn Bush zu führen, um später dann doch auf der Abschussliste zu landen? Der T–800 hat auch keinen Humor, und Menschen sind ja viel zu unzuverlässig, die müssen ersetzt werden. Hacker sind „Cyberterroristen", und Netzaktivisten sowieso, sagt Ex–NSA–Chef Michael Hayden [1]. Ziel dieser Ideologie ist die Kontrolle der Gesellschaft.

Derlei Überwachung gibt es nicht nur im „freien" Westen, sondern natürlich auch in Russland und China (dort ist das nichts Neues), auswandern hilft also nur, wenn man akut von einem der drei Regime verfolgt wird, die anderen beiden aber noch keinen Grund gefunden haben. Es gibt also gute Gründe, einfach alles zu verschlüsseln, also auch alle Katzenfotos und Selfies, und das als Grundverschlüsselung für jedermann zu betreiben. Denn solange nur wenige verschlüsseln, machen die sich automatisch verdächtig.

Net2o hat deshalb keinen Fallback auf nicht verschlüsselte Kommunikation.

Vertraulichkeit, Integrität und Authentizität

Verschlüsselung sorgt nicht nur für Vertraulichkeit (das schützt uns vor dem Lauscher an der Wand), sondern auch für Integrität: die Daten sind unterwegs nicht kaputtgegangen — und Authentizität: die Daten kommen tatsächlich von unserem Kommunikationspartner. Das dient der Robustheit von Kommunikation; die Authentizität nutze ich auch, um ein Handover von einem Netz ins andere mit möglichst geringem Aufwand durchzuführen.

Das Bedrohungsmodell

Was interessiert den Lauscher an der Wand?

- Zunächst einmal, wer mit wem wie oft kommuniziert, die sogenannten „Metadaten"

[1] Qwest

- Dann natürlich die Daten selbst
- Verschlüsselte Daten werden abgespeichert, um sie ggf. später zu entschlüsseln
- Über der Öffentlichkeit als „Porno–Filter“ verkaufte Interception–Router von Huawei können Verbindungen auch mit vergleichsweise teuren Man–in–the–Middle–Attacken (MITM) gezielt manipuliert werden.

Beim „später entschlüsseln“ ist nicht gemeint „wenn wir genügend Rechenleistung haben, um die Entschlüsselung durchzuführen“, sondern „wenn wir den Schlüssel abgegriffen haben“. Denn: Bei Verschlüsselung geht es nicht um absolute Sicherheit, sondern um den Preis, wie man an die Informationen kommt. Der Preis ist nicht unbedingt der, den man sich so vorstellt, weil die Angriffe normalerweise über „Seitenkanäle“ erfolgen, also über Dinge, die man nicht bedacht hat (siehe Abbildung 1). Normalerweise erfolgt der Zugriff auf so einen Schlüssel über einen Trojaner; bei Hackern sind die paranoiden Lauscher allerdings vorsichtig — es könnte sein, dass ihr Trojaner entdeckt, analysiert und damit enttarnt wird. Ein Trojaner von der Stuxnet–Sorte kostet geschätzt mehrere Millionen Dollar, den riskiert man nicht so leichtfertig. Ist natürlich doof, wenn ausgerechnet die Leute, die ganz oben auf der Terrorliste stehen, sich besonders gut wehren können.

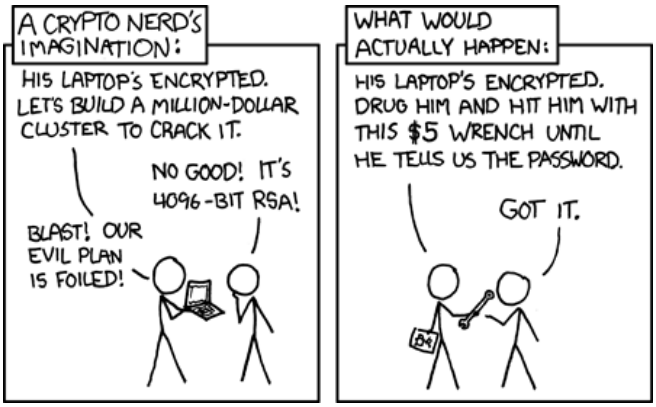


Abbildung 1: xkcd.com/538

Das stellt uns also vor gewisse Herausforderungen, aber nicht vor unlösbare Probleme. So darf beim Verbindungsaufbau die Identität nicht preisgegeben werden — oder zumindest sollten wir erfahren, wenn es einen Versuch gab, sie zu bekommen.

Dann müssen wir natürlich MITM–Attacken verhindern. Dazu ist der Stand der Technik sehr schlecht, der Versuch einer Public–Key–Infrastruktur (PKI) bei SSL und zum Teil auch bei PGP kann als gescheitert angesehen werden. Etwas erfolgreicher ist erstaunlicherweise das SSH–Modell, das lediglich Änderungen anzeigt, und keine PKI enthält. Insbesondere ist bei einer PKI vom PGP–Typ (Web of Trust) bedenklich, dass hier ein „wer kennt wen“–Graph öffentlich ist. Gerade an solchen Graphen sind die Lauscher besonders interessiert.

Verbindungsaufbau

Beim Verbindungsaufbau muss ein Schlüsselaustausch stattfinden, zumindest soweit man noch nicht weiß, mit wem man spricht.

Ich verwende eine Kombination aus ECDHE und ECDH, also einen Elliptic Curve Diffie–Hellman (Ephemeral, also mit nur einmal verwendeten Keys im ersten Schritt) mit curve25519 von Dan Bernstein. Hier erzeugt jede Seite ein Schlüsselpaar sk, pk (secret und public key). Die Operation der elliptischen Kurve wird als Multiplikation bezeichnet ($*$–Operator), auch wenn sich dahinter eine weit komplexere Operation verbirgt. Es gilt aber die Gleichung, dass für zwei Key–Paare $sk1, pk1$ und $sk2, pk2$ die jeweiligen Produkte $sk1 * pk2 = sk2 * pk1$ identisch sind. Da man vom öffentlichen Schlüssel nicht einfach auf den privaten zurückschließen kann, kann das gemeinsame Geheimnis nur gebildet werden, wenn jeder Partner über den jeweiligen geheimen Schlüssel verfügt.

Abbildung 2 zeigt den Verbindungsaufbau. Es werden hier zwei Schlüsselpaare ausgetauscht, das erste ist ein nur einmal verwendeter zufälliger Schlüssel ($sk1, pk1$und $sk2, pk2$). Diese bilden das erste gemeinsame Geheimnis, mit dem dann die eigentlichen, konstanten öffentlichen Schlüssel ausgetauscht werden. Diese Schlüssel stellen die Identität der Teilnehmer dar. Beide Geheimnisse werden zusammengehängt und ergeben den Sitzungsschlüssel für ein symmetrisches Verfahren. Die dicken Pfeile zeigen die tatsächlich übertragenen Daten an. Wir sehen, dass die gemeinsamen Secrets auf beiden Seiten erzeugt werden können, und dass die konstanten public Keys (die Identitäten) von Alice und Bob verschlüsselt übertragen werden.

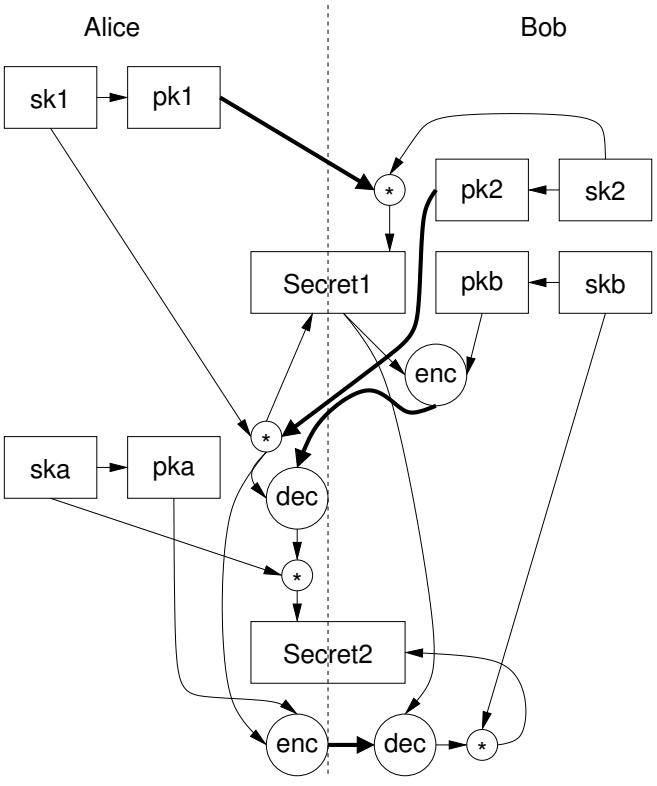


Abbildung 2: Verbindungsaufbau

Ein Angreifer kann in die erste Phase des Schlüsselaustauschs grundsätzlich eingreifen, weil hier zufällige Schlüssel verwendet werden, die ja nicht überprüft werden können. Bei diesem Eingriff ergeben sich aber dann zwei verschiedene Secret1 für beide Seiten, während das Secret2 ja auf beiden Seiten gleich ist, und für den Angreifer nicht ermittelbar (er kennt keinen der beiden geheimen Schlüssel). Damit kann ein Angreifer zwar die Identitäten ermitteln, er wird aber bemerkt, weil der Verbindungsaufbau selbst fehlschlägt.

Glücklicherweise müssen für diesen Verbindungsaufbau nur drei Pakete verschickt werden, also genau die Menge, die man für einen 3–way–Handshake ohnehin braucht. Die Kosten sind symmetrisch: Jede Seite muss ein Schlüsselpaar erzeugen, zweimal Schlüssel multiplizieren und je einmal ver- und entschlüsseln (mit einem symmetrischen Verfahren). Auf einem Core i7 mit 3GHz können etwa 3000 Verbindungen in der Sekunde (pro Kern) aufgebaut werden; das reicht dicke.

Verbindungsaufbau im Detail

Rückblick

Wie im 1. Teil der Folge [2] erklärt, sendet net2o Kommandos hin und her, die eine tokenisierte stackbasierte Sprache benutzen. Da gibt es noch einen kleinen Nachtrag zur Codierung vorzeichenbehafteter Zahlen: Anton Ertl hat den Code etwas verbessert und schlägt vor, die Zickzack–Codierung auch so zu bezeichnen:

```
: zz>n ( zigzag -- n )
  dup 1 rshift swap 1 and negate xor ;
: n>zz ( n -- zigzag )
  dup 0< swap 2* xor ;
```

Wir gucken uns jetzt mal die ausgetauschten Pakete im Detail an, da sieht man dann, was tatsächlich passiert. Die ganzen Strings ersetze ich durch Symbole. net2o kann „nested commands“, das sind Strings, die verschlüsselte Kommandos enthalten — und diese Kommandos werden nach dem Entschlüsseln ausgeführt. Diese Strings werden u.a. auch benutzt, um Zustände zu speichern, die für den Verbindungsaufbau nötig sind. Damit kann man beim Verbindungsaufbau Speicherlecks vermeiden, denn zumindest der angesprochene Teil (im Beispiel oben Bob) muss sich zunächst gar nichts merken.

Initiierung der Verbindung

Das erste Paket von Alice sieht so aus:

```
pk1 $, receive-tmpkey
[: timestamp1 lit, set-rtdelay
   gen-reply request-done
;] $, push-$ push' nest
tmpkey-request key-request
base lit, csize lit, dsize lit, map-request
```

Bei diesen genesteten Kommandos (die ich hier als Quotations notiere) muss man natürlich aufpassen, dass kein Angreifer sie benutzt, um eine Verbindung zu kapern, oder gar Unsinn in den Speicher zu schreiben. Dazu gibt es den Timestamp. Dieser wird als Cookie benutzt und kann nur einmal verwendet werden, und auch nur, solange er gültig ist (und man kann damit den Round–Trip–Delay ausmessen). Der Verbindungsaufbau ist in dieser Phase noch etwas fragil, weil eben noch keine Verschlüsselung ausgehandelt wurde, also auch keine Authentizität besteht. Zudem kann unser angesprochener Kommunikationspartner ja selbst „Späßchen machen,“ also bösartig sein. Damit kein Angreifer massiv jede Menge Cookies schicken kann, um auszuprobieren, ob er 'reinkommt, sind diese Befehle verschlüsselt — mit einem Schlüssel, den nur der Erzeuger des Pakets selbst kennt.

Dieses erste Paket wird mit einem „Generalschlüssel“ verschlüsselt, denn es gibt ja noch keinen ausgehandelten Schlüssel. Es kann damit belauscht werden. Für eine Firewall ist diese Belauschen auch sinnvoll, denn die Daten vor `map-request` entsprechen den Ports bei TCP. Dieses Fenster soll also für eingehende Pakete offen sein. Verändern sollte man es nicht, da man nicht unbedingt weiß, welche Maps der anfordernde Rechner sonst noch verwendet.

Bestätigung des Aufbauwunsches

Von Bob bekommen wir dann folgende Antwort:

```
[: timestamp1 lit, set-rtdelay
   gen-reply request-done ;] $, nest
pk2 $, receive-tmpkey
[: pkb $, receive-key
   [: timestamp2 set-rtdelay
      base' lit, base lit,
      csize lit, new-code
      base'+csize lit, base+csize lit,
      dsize lit, new-data
      secret1 $, store-key
   ;] $, push-$ push' nest
   base lit, base' lit, csize lit, new-code
   base+csize lit, base'+csize lit,
   dsize lit, new-data
;] $, tmpnest
```

Hierbei sendet Bob alle Informationen, die er braucht, um die Verbindung tatsächlich aufzubauen — mit seinem eigenen Nesting–Schlüssel verschlüsselt. Dieses Ticket kann man nur verwenden, um genau diese Verbindung (mit diesen Quell– und Zieladressen) aufzubauen; Bob muss dazu überhaupt nichts speichern, er kann also beliebig viele halboffene Verbindungen ausstehen haben. Das Ticket ist nicht beliebig lang gültig, dafür sorgt der Timestamp. Innerhalb des Timeout–Fensters kann ein Client mehrmals die gleiche Verbindung öffnen, sofern das Mapping noch frei ist. Das ist kein Nachteil, sondern kann als Feature verstanden werden: Für kurzfristiges mehrfaches Aufbauen einer Verbindung zum gleichen Server müssen wir keinen kompletten Schlüsseltausch aushandeln.

Alice sollte noch überprüfen, ob die verwendeten Adressen und Größen passen (Bob kann bei exzessiven Puffergrößen diese verkleinern, da das natürlich auch heißt, dass er sie vergrößern kann, muss Alice das überprüfen).

Verbindungsaufbau

Nun hat Alice alle Informationen, die für eine Verbindung nötig sind, nur Bob noch nicht. Wir schicken also ein drittes Paket auf die Reise:

```
[: timestamp2 set-rtdelay
   base' lit, base lit, csize lit, new-code
   base'+csize lit, base+csize lit,
   dsize lit, new-data
   secret1 $, store-key
;] nest
[: pka $, receive-key update-key
   rnd1 $, gen-code-ivs
   rnd2 $, gen-rcode-ivs
;] tmpnest
```

Wir könnten im Prinzip hier gleich schon weitere Kommandos anhängen, weil zu diesem Zeitpunkt die Verschlüsselung bereits voll ausgehandelt ist; ich empfehle aber, die eigentlichen Kommandos über das nun aufgebaute Command–Mapping zu machen.

Sitzungs–Verschlüsselung

Die eigentlichen Daten werden nach Aushandlung dieses Geheimnisses mit einem symmetrischen Verfahren verschlüsselt. Dabei stehen grundsätzlich mehrere zur Auswahl, weil man ja nie wissen kann, ob nicht eins geknackt wird — für den Fall muss man dann ein anderes wählen. Beim Verbindungsaufbau sieht es etwas trauriger aus: es gibt nicht so viele Verfahren, und unbeschadet hat die Cryptoanalyse nur ECC überstanden. Die anderen Verfahren sind schon noch sicher, aber nur bei sehr langen Schlüsseln.

Ich habe am Anfang Wurstkessel verwendet, weil es kein offiziell abgesegnetes adäquates Verfahren gab, das das tat, was ich brauche. Das hat sich durch den SHA–3–Wettbewerb geändert, bei dem Keccak gewonnen hat. Keccak macht ein paar Dinge etwas anders als Wurstkessel, aber im Kern ist das Konstrukt sehr ähnlich; die Autoren nennen die Modi Sponge und Duplex [3]. Das Konzept ist sorgfältig durchanalysiert, und ermöglicht in einem Durchgang sowohl zu ver- oder entschlüsseln, als auch eine kryptographisch harte Checksumme (einen Hash) zu erzeugen. Oder einen PRNG zu betreiben, bei dem man vom letzten Ergebnis nicht auf das nächste schließen kann. Für den SHA–3–Contest war nur der Hash–Modus relevant, aber selbstverständlich kann man die anderen Modi auch nutzen. Keccak ist schnell (bei vergleichbar paranoider Sicherheit sogar schneller als Wurstkessel), einfach, und weil es durch einen offenen Standardisierungsprozess gegangen ist, auch sehr sicher.

Abbildung 3 zeigt, wie das Verschlüsseln vom Prinzip her aussieht. Es gibt eine Einweg–Funktion f, die den Eingangszustand in einen quasi zufälligen Nachfolgezustand überführt.

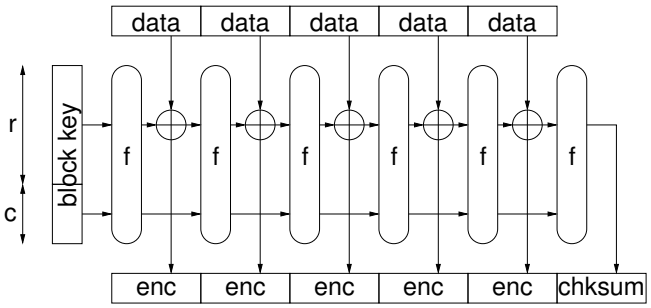


Abbildung 3: Duplex–Mode zur Verschlüsselung

Beim Entschlüsseln geht man umgekehrt vor, siehe Abbildung 4. Es bleibt dabei jeweils nur ein Teil des internen Zustands erhalten. Bei Wurstkessel werden zwei verschiedene Teile des Schlüssels verxodert, was meiner Meinung nach die Sicherheit etwas erhöht; trotzdem ist bei Keccak der unbekannte Teil immer noch 576 Bits pro Schritt, das ist völlig ausreichend.

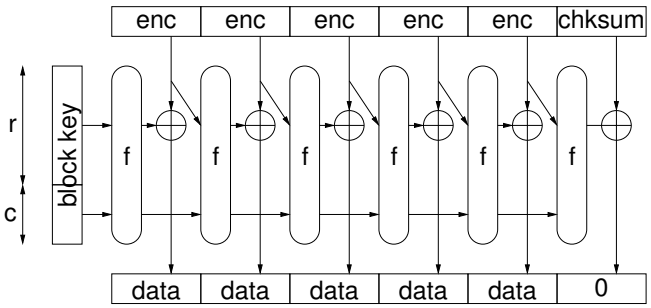


Abbildung 4: Duplex–Mode zur Entschlüsselung

Auf jeden Fall gibt es in net2o die Möglichkeit, verschiedene Algorithmen zu benutzen, solange diese vergleichbare Eigenschaften haben; damit das einfacher zu programmieren ist, benutze ich eine in einer Mini–OOF2 geschriebene Klasse:

```
object class
    method c:init ( -- )
    \G initialize crypto function for a task
    method c:key! ( addr -- )
    \G use addr as key storage
    method c:key@ ( -- addr )
    \G obtain the key storage
    method c:key# ( -- n )
    \G obtain key storage size
    method >c:key ( addr -- )
    \G move 128 bytes from addr to the state
    method c:key> ( addr -- )
    \G get 128 bytes from the state to addr
    method c:diffuse ( -- )
    \G perform a diffuse round
    method c:encrypt ( addr u -- )
    \G Encrypt message in buffer addr u
    method c:decrypt ( addr u -- )
    \G Decrypt message in buffer addr u
    method c:encrypt+auth ( addr u -- )
    \G Encrypt message in buffer addr u
    method c:decrypt+auth ( addr u -- flag )
    \G Decrypt message in buffer addr u
```

```
    method c:hash ( addr u -- )
    \G Hash message in buffer addr u
    method c:prng ( addr u -- )
    \G Fill buffer addr u with PRNG sequence
    method c:checksum ( -- xd )
    \G compute a 128 bit checksum
    method c:cookie ( -- x )
    \G compute a different checksum
end-class crypto
```

Damit kann man den gerade verwendeten Algorithmus jederzeit austauschen, und beim Verbindungsaufbau aushandeln, welchen man nimmt. Allerdings haben die Geheimdienste in der Vergangenheit eine große Auswahl an Algorithmen genutzt, um Verwirrung zu stiften, und Vorschläge zu machen, die die Sicherheit stark beeinträchtigen, wie „Null–Verschlüsselung“ (in IPSEC), Algorithmen mit erheblichen Schwachstellen (AES–CBC und das Padding–Oracle in SSL) oder einfach alte Algorithmen (RC4 in SSL). Zudem sorgt eine große Zahl von Algorithmen dafür, dass es nur wenige Implementierungen gibt, weil es einfach zu mühsam ist.

Viel wichtiger als der Austausch des Algorithmus durch den Endnutzer (der ja ohnehin völlig überfordert ist mit diesem Thema) ist IMHO eine zeitnahe Update–Möglichkeit der Software. Dann kann man Fehler bei der Wahl des Algorithmus oder auch dessen Einsatz überall beheben.

Vorgehensweise

Um symmetrische Verschlüsselung sicher zu benutzen, muss man ein paar Dinge beachten. So darf man jeden Schlüssel nur einmal verwenden; üblicherweise hilft man sich damit, dass man einen zufällig gewählten Initialisierungsvektor (IV) für jedes Paket mitsendet, und mit dem konstanten Sitzungsschlüssel zusammen bildet man damit eine für jedes Paket einmalige Folge von Zuständen im Verschlüsselungsalgorithmus.

Zufall über die Leitung schicken ist aber nicht besonders bandbreitenschonend. Deshalb geht net2o etwas anders vor: Für jede Map (also Code und Daten, auf beiden Seiten) gibt es einen Initialisierungsvektor, der tatsächlich über die Leitung geht. Zusammen mit dem Sitzungsschlüssel, der beim Verbindungsaufbau erzeugt wurde, bilden wir den Status für einen PRNG, einen Zufallsgenerator.

Mit diesem Zufallsgenerator, der auf beiden Seiten synchronisiert ist, erzeugen wir Schlüssel für jedes einzelne Paket, abhängig von dessen Zieladresse. Jedes Mal, wenn der Puffer voll ist, und wir einen Rewind durchführen, erzeugen wir einen neuen Satz Schlüssel.

Diese Vorgehensweise hat folgende Vorteile:

- Zum einen müssen wir den IV nicht mit jedem Paket mitsenden und sparen so Bandbreite
- Zum anderen gibt es auf keiner Seite einen konstanten Sitzungs–Schlüssel im Speicher, den man auslesen kann, und damit den ganzen Kommunikationsvorgang noch im Nachhinein entschlüsseln kann. Alles ist nur so lang im Speicher, so lange es gebraucht wird. Da wir die temporären asymmetrischen Schlüssel nach erfolgreichem Verbindungsaufbau wegwerfen können, stimmt das auch tatsächlich.

Der Betrieb von Keccak oder Wurstkessel im Duplex–Modus bedeutet, dass am Anfang der Paket–Schlüssel in den Zustands–Speicher des Algorithmus geladen wird, dann blockweise per XOR die Message verschlüsselt und in den Zustand aufgenommen wird, und über die Diffusionsfunktion ein völlig anderer, nicht erkennbar abhängiger Zustand produziert wird. Dabei wird in jedem dieser XOR–Schritte nur ein Teil des internen Zustands verändert, der andere Teil bleibt für einen Angreifer in jedem Fall sowohl unbekannt als auch unberührt.

Am Ende ist dann ein Hash im Zustandsspeicher, der sowohl vom anfangs gewählten Schlüssel als auch von der Message abhängt. Von diesem Hash hänge ich 128 Bits als Authentifizierung an das Paket an, und speichere 64 weitere Bits als „Cookie“ für diese Adresse (die Speicher–Adresse, net2o–Pakete übertragen ja Daten zwischen gemeinsamen Speichern). Abbildung 5 zeigt den Schlüsselfluss. Die Cookies werden verwendet, um zu überprüfen, ob Acknowledges legitim sind, und sich auf tatsächlich empfangene Daten beziehen. Acknowledges werden ja summarisch übermittelt, es gibt eine Startadresse und eine Bitmaske der empfangenen Pakete, die dabei erhaltenen Cookies werden verxodert.

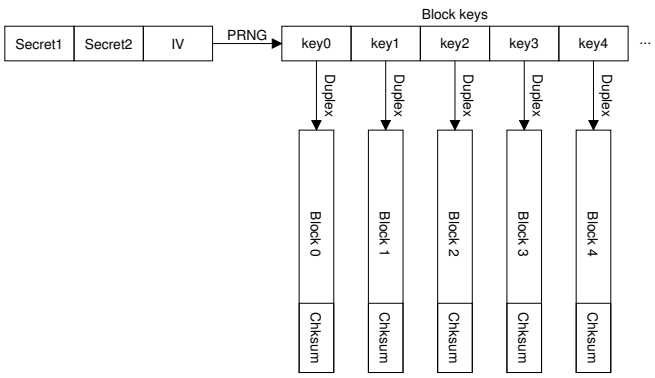


Abbildung 5: „Schlüsselfluss“

Der Empfänger führt nur leicht abgewandelt dieselben Schritte durch, und erhält dabei die entschlüsselte Message sowie einen Hash zum Vergleichen. Ist die Authentizität belegt, kommt also unser Paket von jemandem, der über den Schlüssel verfügen muss (weil er sonst keinen korrekten Hash generieren kann), ist auch die Integrität des Pakets belegt: Es ist offensichtlich kein einziges Bit gekippt. Das beinhaltet auch die Adresse, mit der ja der richtige Schlüssel gewählt wurde.

Wenn der Hash nicht stimmt, verwerfen wir das Paket. Eine Antwort ist nicht nötig, wir können nämlich nicht wissen, ob das Paket vom Sender kam, oder von einem

anderen auf den Weg geschickt wurde — eine Fehlermeldung zurück an den vermeintlichen Absender schadet also mehr, als sie nützt.

Zufall

Eins der wichtigsten Dinge bei der Kryptographie ist zuverlässiger Zufall. Zuverlässig ist Zufall dann, wenn man keinerlei Gesetzmäßigkeiten erkennen kann, außer den Gesetzen der großen Zahl und der stochastischen Gleichverteilung. Ich nutze den Zufall, um private Schlüssel zu generieren, und um Initialisierungsvektoren (IV) zu erzeugen. Schwacher Zufall ist ziemlich schlecht... Woher kriegen wir den Zufall?

Unter Linux gibt es zwei Dateien, die uns Zufall liefern: `/dev/random` und `/dev/urandom`. Erstere liefert nur so viele Bits, wie der Kernel glaubt, an zufälliger Entropie gewonnen zu haben, letztere liefert immer etwas, auch wenn es nicht garantiert ist, wie zufällig das ist. Das alles klingt nach „naja," und ist es irgendwo auch, denn der Kernel kann nicht garantieren, dass er auch Quellen für Zufall hat (in der MIPS–Version von Linux liefert `get_cycle()` immer 0, obwohl das die am häufigsten genutzte Entropiequelle von `/dev/random` ist — das betrifft die ganzen Router, bei denen MIPS sehr beliebt ist). In den neueren Core–Prozessoren hat Intel eine recht flotte Zufallsquelle eingebaut (echtes Rauschen, kein Pseudozufall), aber erst seit Haswell ist das „NIST–konform". Also hat alle Tests bestanden, die das NIST für solche Zufälle haben will.

Ich gehe deshalb wie folgt vor: Jeder User hat eine Datei `.initrng` in seinem Homeverzeichnis. Ist die leer oder noch nicht da, dann nutzen wir `/dev/random`, um sie zu befüllen. Ansonsten ist das unser initialer Schlüssel für Keccak, den wir einmal durch den Wolf drehen (`c:diffuse`). Was verschlüsseln wir? Jeweils ein Kilobyte Daten von `/dev/urandom`. Diese Daten enthalten hoffentlich ohnehin eine Menge Zufall, nach der Verschlüsselung sind wir sicher, dass sie allen notwendigen Kriterien (also Gleichverteilung und so) genügt. Nachdem wir ein Kilobyte so erfolgreich verschlüsselt haben, und als Vorrat für Zufallszahlen nutzen, schreiben wir den Schlüssel wieder in die `.initrng` 'raus, und verschlüsseln noch ein Kilobyte, denn das erste war nur da, damit der nächste Prozess einen Startwert bekommt, den noch keiner gesehen hat. Der wird den Schlüssel mit `c:diffuse` und einer neuen Runde für den nächsten Startwert modifizieren, wir nicht, wir generieren von diesem Punkt aus unsere weiteren Zufallszahlen.

Worauf muss man sonst noch achten? Wenn sich der Prozess oder der Thread geändert hat, muss der Random–Generator neu initialisiert werden. Der Random–Generator ist nicht thread–safe, besser, man hat eine eigene Kopie für jeden Thread. Und wenn ein fork() durchgeführt wird (dann ändert sich die pid), muss auf jeden Fall auch neu initialisiert werden — sonst arbeiten zwei Prozesse zumindest zeitweise mit den gleichen Zufallsdaten.

Identität und Vertrauen

Der letzte Abschnitt dreht sich darum, wie man MITM–Attacken abwehrt. Das Angriffsmodell eines Diffie–Hellman–Schlüsselaustausches sieht eigentlich nur einen passiven Lauscher vor, nicht einen, der aktiv in die Leitung eingreift. Wenn sich Eve zwischen Alice und Bob klemmt, und sich gegenüber Alice als Bob ausgibt und umgekehrt, kann sie jeweils eine verschlüsselte Verbindung zu beiden aufbauen, aber dazwischen alles abhören.

Was macht man dagegen? Alice und Bob müssen irgendwie vertrauen, dass die Schlüssel, die ausgetauscht werden, auch tatsächlich zu Alice und Bob gehören. Wie geht das? Und wie vermeidet man, dass sich Alice und Bob zu diesem Zweck persönlich begegnen müssen — denn wenn das nötig ist, könnten sie sich auch gleich ein „shared secret" vereinbaren.

Die traditionelle Vorgehensweise bei SSL und PGP ist, diese Prüfung der Identität an jemanden zu delegieren. Bei SSL ist es eine Certificate Authority (CA) genannte Instanz, von denen es etwa 600 gibt, davon so vertrauenswürdige wie DigiNotar, GoDaddy, Comodo, Türktrust, CNNIC (da steckt die chinesische Regierung dahinter) oder die Deutsche Telekom und AOL. Bei DigiNotar ist jemand eingebrochen, sie sind sogar offiziell nicht mehr vertrauenswürdig. Aber der Rest? Jeder dieser CAs kann Zertifikate für jede Website ausstellen. Das System kann als komplett gebrochen angesehen werden. Wenn bei den großen etwas schief geht (GoDaddy, Comodo, Verisign), dann sind die „too big to fail", und in der Tat ist auch bei denen etwas schief gegangen.

Ist PGP jetzt besser? Bei den Key–Signing–Parties kann man begutachten, ob die anderen Teilnehmer ihre Aufgabe ernst nehmen; in der Regel führt man sie dann als „marginally trusted". Davon braucht es drei, um einem weiteren zu vertrauen. Also muss die NSA drei Nerds auf so eine Veranstaltung schicken, und kann dann für jeden Teilnehmer einen vertrauenswürdigen Key erzeugen. Das sollten die wohl hinbekommen, wenn es Profis wären. Auch dieses Modell taugt also nichts. Insbesondere ist die Key–Signing–Party so kompliziert, dass auch bei den Teilnehmern z.B. am LinuxTag 2/3 es nicht schaffen, die Keys anschließend tatsächlich zu unterschreiben und an den Rest der Teilnehmer zu verschicken. Für Normalmenschen ist diese Vorgehensweise völlig ungeeignet. Und: Die Vorgehensweise macht etwas, was man u.U. überhaupt nicht will: Sie ordnet die Schlüssel offiziellen Dokumenten zu. Gerade, wenn man unerkannt bleiben will, ist das eine sehr blöde Idee.

SSH verzichtet ganz auf derartige Signaturen. Hier wird einfach jede Änderung des Schlüssels angezeigt. Schlüssel ändern sich nicht so mir nichts, dir nichts ohne Grund. Die Verwendung des gleichen Schlüssels bei der nächsten Sitzung ist ein Vertrauensbeweis. Das schützt vor gelegentlichen Angriffsversuchen, aber nicht in einem sogenannten „captive environment". Also zum Beispiel innerhalb einer Firma, deren Firewall grundsätzlich alle Verbindungen beschnüffelt.

Andere Lösungsmöglichkeiten

Je nach Anwendungsfall gibt es alternative Lösungsmöglichkeiten. Ein typischer Anwendungsfall verschlüsselter Kommunikation ist das Anmelden bei einem Server. Man benutzt die verschlüsselte Übertragung, um das Passwort über die Leitung zu schicken. Das ist hier überhaupt nicht nötig, beide Teilnehmer, Alice und Bob, authentisieren sich mit ihrem geheimen Schlüssel, und identifizieren sich mit ihrem öffentlichen Schlüssel. Das Merken des öffentlichen Schlüssels auf beiden Seiten reicht also völlig aus, es reicht, beim ersten Verbindungsaufbau nicht belauscht zu werden; das ist weitgehend sicher, außer man ist in besagtem „captive environment."

Dann ist da noch die Frage der Identität, oder: wie erreiche ich jemanden. Traditionell bei e-Mail sind Namen oder Pseudonyme und eine Domain, die die Mails in Empfang nimmt. Der Schlüssel hat damit nichts zu tun. Meine Idee ist hier, den öffentlichen Schlüssel direkt als Identität zu verwenden. Ein Name, ein Pseudonym, das kann man verwenden, um den Schlüssel zu finden, aber auf die Visitenkarte muss nur der Schlüssel selbst. Dabei muss man den Schlüssel nicht unbedingt voll ausschreiben, ein Teil davon reicht. Mit Nummern können auch Normalmenschen umgehen, selbst am Handy lassen sich Nummern schnell und einfach eingeben. Die Frage nach einem geeigneten Pseudonym stellt sich nicht, man bekommt gleich eine zufällige Nummer. Das löst das Problem der Schlüsselübergabe bei direktem Kontakt zweier Normalmenschen, die sich nicht mit Kryptographie auseinandersetzen wollen. Man muss die Identität eines Menschen nicht überprüfen, wenn jemand sagt „das ist mein Schlüssel", dann ist das sein Schlüssel — eine Kommunikation kommt nur zustande, wenn das auch stimmt. Ob Alice wirklich Alice heißt, und Bob wirklich Bob, ob der Schlüssel evtl. einer ganzen Gruppe gehört, oder ob sie diese Namen einfach nur angenommen haben, um im Online–Chat einen Seitensprung zu wagen (Eve hat schon einen Grund, warum sie Bob hinterherschnüffelt), das ist ihre Sache.

Nun, was ist mit Kontakten, die nicht persönlich hergestellt werden? Oder Kontakte mit Rechnern, die sich eh nicht persönlich vorstellen? Eine Möglichkeit ist die von Captchas. Hier überprüft der Rechner, ob der am anderen Ende ein Mensch ist, und das gleiche Geheimnis für die Verbindung nutzt. Dazu muss der Mensch z.B. verzerrte Buchstaben und Zahlen erkennen — und dann aus dem gemeinsamen Geheimnis in Matrixdarstellung die passenden Ziffern eintippen. Es geht hier nur darum, den Aufwand für eine abgefangene Verbindung so hoch zu machen, dass es sich nicht lohnt, das massenhaft durchzuführen. Für eine erstmalige Anmeldung bei einem Server müssen die Leute auch heute schon Captchas lösen; das sind sie gewohnt, der Aufwand ist nicht groß. Wenn zwei Menschen über Sprache verbunden sind, können sie sich die Zahlen des Geheimnisses auch gegenseitig vorlesen — das müssen nicht alle sein. Auch das Vorlesen des Public Keys ist hilfreich. Bei einem Video gibt es auch die Möglichkeit, abhängig vom Geheimnis Teile auszumaskieren und durch eine einzige Farbe zu ersetzen. Die andere Seite kann dann die Umrisse der zu erwartenden Maskierung darüber abbilden. Innerhalb darf nur die Masken–Farbe zu sehen sein, außerhalb das Bild des Gegenübers. Wenn zwei Rechner miteinander kommunizieren, geht das natürlich nicht. Deshalb sollte man alle Nachrichten, die über mehrere Rechner wandern, Ende–zu–Ende verschlüsseln. Am Ende sitzt dann eben ein Mensch, und der kann mehr überprüfen.

Beim Aufbau eines Freundeskreises in einem sozialen Netzwerk nutzt man seine Freunde als Mittler, also als Vertrauenspersonen. Wenn die Freunde diesen Key schon kennen, dann ist es entweder ein Agent, der die ganze Gruppe unterwandern will, oder tatsächlich ein Freund. Beides ist mit krytographischen Mitteln nicht zu unterscheiden. Das gilt auch für Sites, die man anderen empfiehlt — wenn man hier den Key bekannt gibt, hilft das den Freunden beim Verbindungsaufbau.

Das sind alles Möglichkeiten, um die Vertrauenswürdigkeit der Verbindung zu testen, ohne dass man dazu die Identität feststellen muss, und ohne, dass man diese Feststellung an jemand anderen delegieren muss, dessen Vertrauenswürdigkeit man nicht überprüfen kann. Denn leider erzeugt ein Schnüffelstaat Paranoia, also einen generellen Vertrauensverlust.

Sichererer als Internet

Letztendlich ist das ganze Fangen von richtigen Terroristen so schwierig, weil die gar keine Telefone oder Computer benutzen, um miteinander zu kommunizieren, sondern Boten und tote Briefkästen und so. Das führt dann auch zu einer schönen Ausweitung des Verdachtsmoments: Wenn man einen Terrorist kennt, dann kann jeder, der mit dem in Kontakt kommt, ein Bote sein (1 Hop). Der läuft dann zum nächsten Terrorist (2 Hops). Und vielleicht benutzt der andere Terrorist ja auch nur Boten, um mit der Außenwelt in Kontakt zu geraten (3 Hops). Gleiches gilt für Gebäude. Die Mafia macht das seit Jahrhunderten so. Das ist ziemlich sicher, denn mit 3 Hops erreicht man so viele Leute, dass sich die tatsächlichen Kriminellen in der Masse der falschen Verdächtigen verlieren.

Die ganzen US–Geheimdienste haben ein Budget von 10 Milliarden Dollar allein für Kryptographie und beschäftigen damit 35 000 Leute, nur um irgendeinen möglichen signifikanten Durchbruch in der Kryptanalyse zu erzielen. Das dürfen aber zum Glück alles keine brillanten Leute sein, weil die einem Geheimdienst nur Ärger machen. Also geht von denen keine große Gefahr aus.

Als Listings gibt's dieses Mal den Zufallsgenerator und eine einfache Implementierung von Keccak für 64–Bit–Systeme. Die ist einfach, aber lange nicht so performant wie die C–Implementierung der Erfinder. Aber man kann an ihr sehen, was Keccak eigentlich macht.

Und was noch fehlt, das ist der Austausch von Nachrichten, die irgendwo zwischengespeichert werden, wenn man nicht online ist — das kommt aber ein anderes Mal.

Literaturverzeichnis

[1] The Guardian, *Former NSA chief warns of cyber-terror attacks if Snowden apprehended*, http://www.theguardian.com/technology/2013/aug/06/nsa-director-cyber-terrorism-snowden

[2] Bernd Paysan, *net2o — Das Internet neu erfinden, Teil 1,* VD 2012/04

[3] Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, *The Sponge Functions Corner,* http://sponge.noekeon.org/

[4] Bernd Paysan, *net2o fossil repository,* https://fossil.net2o.de/net2o

Listings

```
\ generic rng

require unix/pthread.fs

$400 Constant rngbuf#

s" /dev/urandom" r/o open-file throw Value rng-fd

User rng-pos
User rng-buffer
User rng-pid
User rng-task
User rng-key
rngbuf# rng-pos !

: rng-allot ( -- )
    rngbuf# allocate throw rng-buffer !
    c:key# allocate throw rng-key !
    rngbuf# rng-pos !
    getpid rng-pid ! up@ rng-task ! ;

: rng-exec ( xt -- )  c:key@ >r  rng-key @ c:key!
    execute  r> c:key! ;

: rng-init ( -- )
    rng-buffer @ rngbuf# rng-fd read-file throw drop ;

: rng-step ( -- )
    [: rng-init
       rng-buffer @ rngbuf# c:encrypt
       rng-pos off ;] rng-exec ;

\ init rng to be actually useful

: random-init ( -- )
    s" /dev/random" r/o open-file throw >r
    rng-key @ c:key# r@ read-file throw drop
    r> close-file throw ;

: read-initrng ( fd -- flag )  { fd }
    0. fd reposition-file throw
    rng-key @ c:key# fd read-file throw c:key# =
    c:diffuse  fd close-file throw ;

: write-initrng ( -- )
    s" ~/.initrng" r/w create-file throw >r
    rng-key @ c:key# r@ write-file throw
    r> close-file throw ;

: salt-init ( -- )
    s" ~/.initrng" r/o open-file IF  drop random-init
    ELSE  read-initrng  0= IF  random-init  THEN  THEN
    rng-step write-initrng rng-step ;

\ buffered random numbers to output 64 bit at a time

: rng-step? ( n -- )
    up@ rng-task @ <> getpid rng-pid @ <> or
    IF  rng-allot salt-init  THEN
    rngbuf# u> IF  rng-step  THEN ;

: rng@ ( -- x )
    rng-pos @ 64aligned 64'+ rng-step?
    rng-pos @ 64aligned dup 64'+ rng-pos !
    rng-buffer @ + 64@ ;

: rng$ ( u -- addr u ) >r
    rng-pos @ r@ + rng-step?
    rng-buffer @ rng-pos @ + r> dup rng-pos +! ;

: rng32 ( -- x )
    rng-pos @ 4 + rng-step?
    rng-pos @ rng-buffer @ + l@
    4 rng-pos +! ;
```

```
\ Keccak: Forth version by Bernd Paysan
\ derived from "readable keccak"
\ 19-Nov-11  Markku-Juhani O. Saarinen <mjos@iki.fi>
\ A baseline Keccak (3rd round) implementation.

24 Value keccak-rounds
5 cells constant kcol#
25 cells constant kkey#

: carray  Create  DOES> + c@ ;
: array   Create  DOES> swap cells + @ ;

array keccakf-rndc
$0000000000000001 , $0000000000008082 ,
$800000000000808a , $8000000080008000 ,
$000000000000808b , $0000000080000001 ,
$8000000080008081 , $8000000000008009 ,
$000000000000008a , $0000000000000088 ,
$0000000080008009 , $000000008000000a ,
$000000008000808b , $800000000000008b ,
$8000000000008089 , $8000000000008003 ,
$8000000000008002 , $8000000000000080 ,
$000000000000800a , $800000008000000a ,
$8000000080008081 , $8000000000008080 ,
$0000000080000001 , $8000000080008008 ,

carray keccakf-rotc
1 c,  3 c,  6 c,  10 c, 15 c, 21 c,
28 c, 36 c, 45 c, 55 c, 2 c,  14 c,
27 c, 41 c, 56 c, 8 c,  25 c, 43 c,
62 c, 18 c, 39 c, 61 c, 20 c, 44 c,

: cc,  cells c, ;

carray keccakf-piln
10 cc, 7 cc,  11 cc, 17 cc, 18 cc, 3 cc,
5 cc,  16 cc, 8 cc,  21 cc, 24 cc, 4 cc,
15 cc, 23 cc, 19 cc, 13 cc, 12 cc, 2 cc,
20 cc, 14 cc, 22 cc, 9 cc,  6 cc,  1 cc,

carray mod5
0 cc, 1 cc, 2 cc, 3 cc, 4 cc,
0 cc, 1 cc, 2 cc, 3 cc, 4 cc,

\ update the state with given number of rounds

kcol# buffer: bc
kkey# buffer: st

: lrot1 ( x1 -- x2 )  dup 2* swap 0< - ;
: lrot ( x1 n -- x2 )  2dup lshift >r
    64 swap - rshift r> or ;
: xor! ( x addr -- )  dup >r @ xor r> ! ;

: theta1 ( -- )
```

```
56      5 0 DO
57          0 st i cells + kkey# bounds DO
58          I @ xor  kcol# +LOOP
59          bc i cells + !
60      LOOP ;
61
62  : theta2 ( -- )
63      5 0 DO
64          bc I 4 + mod5 + @
65          bc I 1 + mod5 + @ lrot1 xor
66          st i cells + kkey# bounds DO
67          dup I xor!  kcol# +LOOP
68          drop
69      LOOP ;
70
71  : rhopi ( -- )
72      st cell+ @
73      24 0 DO
74          I keccakf-piln st + dup @
75          rot I keccakf-rotc lrot
76          rot !
77      LOOP drop ;
78
79  : chi ( -- )
80      st kkey# bounds DO
81          I bc kcol# move
82          5 0 DO
83              bc I 1+ mod5 + @ invert
84              bc I 2 + mod5 + @ and
85              J I cells + xor!
86          LOOP
87      kcol# +LOOP ;
88
89  : iota ( round -- )
90      keccakf-rndc st xor! ;
91
92  : oneround ( round -- )
93      theta1  theta2  rhopi  chi  iota ;
94
95  : keccakf ( -- )
96      keccak-rounds 0 ?DO  I oneround  LOOP ;
97
```

```
98  : st0 ( -- )  st kkey# erase ;
99
100 : >sponge ( addr u -- )
101     \ fill in sponge function
102     st swap bounds DO
103         dup @ I xor!  cell+
104     cell +LOOP  drop ;
105
106 : >duplex ( addr u -- )
107     \ duplex in sponge function: encrypt
108     st swap bounds DO
109         dup @ I @ xor dup I ! over !  cell+
110     cell +LOOP drop ;
111
112 : duplex> ( addr u -- )
113     \ duplex out sponge function: decrypt
114     st swap bounds DO
115         dup @ I @ xor over @ I ! over !  cell+
116     cell +LOOP drop ;
117
118 \ for test, we pad with Keccak's padding function
119
120 144 buffer: kpad
121
122 : padded>sponge ( addr u1 u2 -- )  >r
123     \ pad last round
124     kpad r@ erase  tuck kpad swap move
125     kpad + 1 swap c!
126     kpad r@ + 1- dup c@ $80 or swap c!
127     kpad r> >sponge  ;
128
129 0 [IF]    ." Test "
130     \ tests - we check only for the first 64 bit
131     \ but repeat keccakf 4 times. The input pattern is
132     \ from an official Keccak test, the output as well.
133     st0 s" SX{9" $80 padded>sponge 0 st 4 + c!
134     keccakf st @ $466624B803BF072F =
135     keccakf st @ $993340D7F9153F02 = and
136     keccakf st @ $6EAAAE36BE8E36D3 = and
137     keccakf st @ $1B4AEC08DA6A8BA6 = and
138     [IF] ." succeeded" [ELSE] ." failed" [THEN] cr
139 [THEN]
```

Abbildung 6: Immer noch nicht entschlüsselt: Diskos von Phaistos (Bild: Wikipedia)

SPI — SRAM am GP32

Rafael Deliano

Für viele Applikationen reicht das RAM im GP32 nicht aus. Ein 32–kByte–SRAM, das über SPI angesteuert wird **(Abb. 4)**, bietet mehr als genug Speicher und belegt kaum Portpins. Es ist aber natürlich deutlich langsamer als internes RAM. Das IC von Microchip ist über ebay aus USA in Kleinmengen als SO8 oder DIL8 beschaffbar. Da der GP32 normalerweise mit 5V betrieben wird, das SRAM aber mit 3,3V, ist ein Adapter mit Pegelwandler erforderlich, der auch einen passenden Spannungsregler enthält **(Abb. 1 und 5)**. Für manche Anwendung sind 64 kByte SRAM günstiger. Das Board wurde deshalb auf zwei Speicher-ICs ausgelegt.

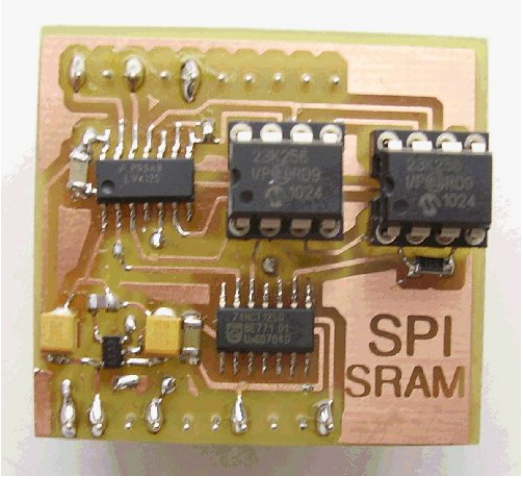


Abbildung 1: Modul

SPI

Es ist eine traurige Erfahrungstatsache, dass low-level Treiber für SPI in Assembler recht bald zufriedenstellend funktionieren, während man bei der Verwendung der Peripherieschaltung des GP32 meist eine zeitraubende Fehlersuche vor sich hat. Da man beim Zugriff aufs SRAM aber hohe Geschwindigkeit anstrebt, will man hier doch die I/O des Controllers verwenden.

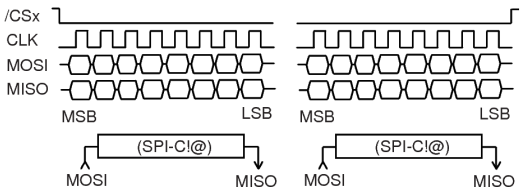


Abbildung 2: SPI-Timing

Eine Fehlerquelle ist der /CS-Pin, der als Portpin ausgeführt ist, und damit nicht ins Timing der GP32-Schaltung eingebunden ist. Er darf erst auf high schalten, wenn tatsächlich alle Bits gesendet wurden. Obwohl das SRAM halbduplex angesteuert wird, wird man also mit Routinen wie `(SPI-C!)` und `(SPI-C@)` nicht glücklich. Erfolgversprechend ist ein Befehl `(SPI-C!@)` der ein Byte sendet, wartet, ein Byte empfängt. In Listing–1 ist sowohl die Version für die I/O–Schaltung als auch die äquivalente Assemblervariante dargestellt. Listing–2 enthält als Beispiel für Anwendung den Lesezugriff aufs SRAM.

FORTH

Die Einbindung ist angenehm einfach, File mit Patch compilieren genügt[1]. Da die untere Hälfte der GP32 Memory Map ohnehin kaum belegt ist **(Abb. 3)**, kann man das RAM dort virtuell einfügen. Man definiert die Befehle `@ ! C@ C! B@ B1! B0!` neu. Bei Adressen im Bereich 0400 ... 7FFF wird der SPI-Zugriff ausgelöst, sonst Ausführung der bisherigen Befehle in nanoFORTH. Hat man diese Nucleus–Befehle, compiliert man in Hochsprache formulierte Versionen von `FILL` , `CMOVE` und `DUMP`.

Da die wichtigen Befehle `@ ! C@ C!` durch diese Verzweigung natürlich etwas langsamer werden, sollten alle zeitkritischen Teile einer Applikation, die nicht auf das externe RAM zugreifen, vorher compiliert worden sein.

Neue 5V–Typen

Nur über Digikey beschaffbar sind die 23LC1024 mit üppigem 128k x 8 Speicher. Was natürlich für Datenlogger interessant ist. Und zwar nicht nur in DIL, sondern auch für 5V–Betriebsspannung, wie sie Automotiv–Großkunden weiter fordern.

Links

`http://www.embeddedforth.de/`

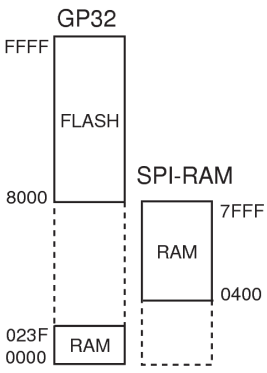


Abbildung 3: Memory Map 32kSRAM

[1] Gemeint ist nanoForth für den GP32

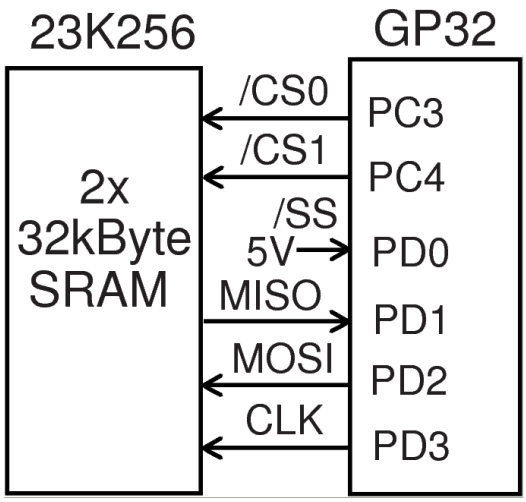


Abbildung 4: Blockschaltbild

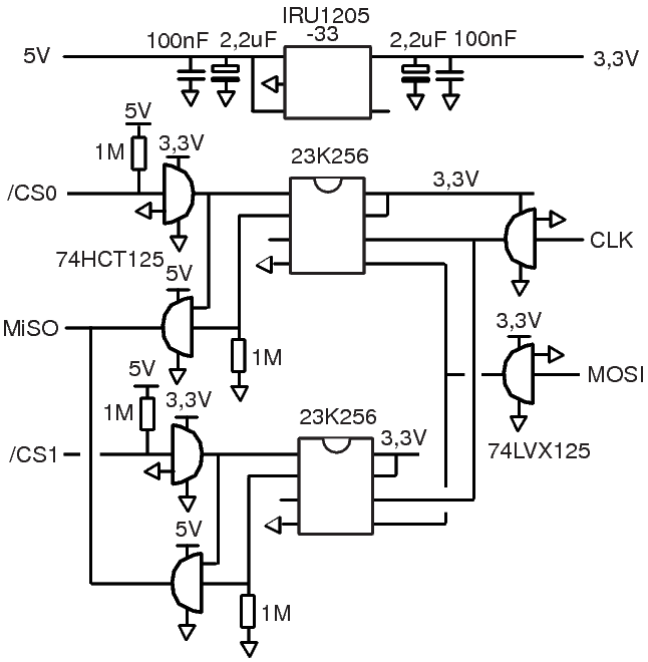


Abbildung 5: Schaltung

Listing

```
1   Listing 1: IO-Routine
2   \ Version Assembler
3   :CODE (SPI-C!@)           \ in:  akku = data
4                             \ out: akku = data
5
6         N 1+ 08 #. MOV,     \ number of bits
7         A.         ROL,
8   2 $:
9   1 $              BCS,
10        MOSI       MBC,
11  3 $              BRA,
12  1 $:  MOSI       MBS,
13  3 $:  SCK        MBS,
14  4 $   MISO       BBS,     \ sets carry
15  4 $:  A.         ROL,
16        SCK        MBC,
17        N 1+       DEC,
18  2 $              BNE,
19                   RTS,
20  CODE;
21
22  \ Version SPI-Schaltung GP32
23  :CODE (SPI-C!@)       \ in:  akku = data
24                        \ out: akku = data
25  1 $: 1 $ 0011 3 BBC,  \ wait for transmitter empty
26           0012   STA,
27           0012   LDA,  \ clear flag
28  2 $: 2 $ 0011 7 BBC,  \ wait for receiver full
29           0012   LDA,
30                  RTS,
31  CODE;
32
```

```
Listing 2: Lesezugriff SRAM
:CODE SPI-C@        \ ( addr --- C1 ) read SPI-SRAM
                    \ RAM in default byte-mode assumed
     /CS-SRAM         MBC,
                03 #. LDA,   \ READ-Token
' (SPI-C!@)           JSR,
                  0,X LDA,   \ Addr HB
' (SPI-C!@)           JSR,
                  0,X CLR,
                 1 ,X LDA,   \ Addr LB
' (SPI-C!@)           JSR,
' (SPI-C!@)           JSR,
                 1 ,X STA,   \ data C1
     /CS-SRAM         MBS,
                      RTS,
CODE;
```

(In der folgenden Forth-Quelle ist exemplarisch gezeigt, wie das serielle Ram transparent im Forth eingebunden werden kann, und wie Forth-Worte auch im seriellen Ram operieren können. Zur Unterscheidung von Operationen im „ echten“ Ram benutzt der Autor den Begriff *Token*, und das `T` als Präfix.

```
\ SRC20

PD 1 DCONSTANT MISO  \ in
PD 2 DCONSTANT MOSI  \ out
PD 3 DCONSTANT SCK   \ out

PC 3 DCONSTANT /CS-SRAM1  \ out
PC 4 DCONSTANT /CS-SRAM0  \ out

:CODE INIT                \ ( ---  )
PC 3 MBS,    DPC 3 MBS,
PC 4 MBS,    DPC 4 MBS,

 MOSI MBC, DPD 2 MBS,
  SCK MBC, DPD 3 MBS,
RTS,
CODE;

:CODE (SPI-C!@)           \ in:  akku = data
                          \ out: akku = data

      N 1+ 08 #. MOV,     \ number of bits
      A.         ROL,
2 $:
1 $              BCS,
      MOSI       MBC,
3 $              BRA,
1 $:  MOSI       MBS,
3 $:  SCK        MBS,
4 $   MISO       BBS,     \ sets carry
4 $:  A.         ROL,
      SCK        MBC,
      N 1+       DEC,
2 $              BNE,
                 RTS,
CODE;

\ ------------------------------------------------------------

:CODE TC@                   \ ( addr --- C1 )
```

```
41                                  \ RAM in default byte-mode assumed
42                      0,X LDA,
43   1 $                    BMI,
44        /CS-SRAM0         MBC,
45   2 $                    BRA,
46   1 $: /CS-SRAM1         MBC,
47   2 $:
48                    03 #. LDA,    \ READ-Token
49   ' (SPI-C!@)            JSR,
50                      0,X LDA,
51   ' (SPI-C!@)            JSR,
52                      0,X CLR,
53                     1 ,X LDA,
54   ' (SPI-C!@)            JSR,
55   ' (SPI-C!@)            JSR,
56                     1 ,X STA,
57        /CS-SRAM0         MBS,
58        /CS-SRAM1         MBS,
59                         RTS,
60   CODE;
61
62   :CODE C@                           \ ( addr --- C1 )
63                      0,X LDA,
64   1 $                    BMI,
65           B% 11111100 #. AND,
66   1 $                    BEQ,
67   ' SPI-C@               JMP,
68   1 $:
69   ' C@                   JMP,
70   CODE;
71
72   :CODE TC!                          \ ( C1 addr ---  )
73                                      \ RAM in default byte-mode assumed
74                      0,X LDA,
75   1 $                    BMI,
76        /CS-SRAM0         MBC,
77   2 $                    BRA,
78   1 $: /CS-SRAM1         MBC,
79   2 $:
80   SOLVE$
81                    02 #. LDA,    \ WRITE-Token
82   ' (SPI-C!@)            JSR,
83                      0,X LDA,
84   ' (SPI-C!@)            JSR,    \ addr HB
85                     1 ,X LDA,
86   ' (SPI-C!@)            JSR,    \ addr LB
87                     3 ,X LDA,
88   ' (SPI-C!@)            JSR,    \ data C1
89        /CS-SRAM0         MBS,
90        /CS-SRAM1         MBS,
91          INX, INX, INX, INX,
92                         RTS,
93   CODE;
94
95   :CODE C!                           \ ( C1 addr --- )
96                      0,X LDA,
97   1 $                    BMI,
98           B% 11111100 #. AND,
99   1 $                    BEQ,
100  ' SPI-C!               JMP,
```

```
1 $:
' C!                    JMP,
CODE;


: T!                          \ ( UN1 addr --- ) bigendian
2DUP SWAP 8SHIFT> SWAP TC! 1+ TC! ;

\ : T!                        \ ( UN1 addr --- ) littleendian
\ 2DUP TC! SWAP 8SHIFT> SWAP 1+ TC! ;

: T@                          \ ( addr --- UN1 ) bigendian
DUP TC@ 8<SHIFT SWAP 1+ TC@ OR ;

\ : T@                        \ ( addr --- UN1 ) littleendian
\ DUP TC@ SWAP 1+ TC@ 8<SHIFT OR ;

." -> SRC21"
|>
<| \ SRC21

HEX

\ ------------------------------------------------------------

: TB@   \ ( addr C# --- Flag )  C# = 0 ... 7
BS-TAB + C@ SWAP TC@ AND ;

: TB1!   \ ( addr C# ---  )  C# = 0 ... 7
BS-TAB + C@ OVER TC@ OR SWAP TC! ;

: TB0!   \ ( addr C# ---  )  C# = 0 ... 7
BS-TAB + C@ NOT OVER TC@ AND SWAP TC! ;

: TCMOVE                      \ ( Addr1 Addr2 N1  ---     )
                              \ Addr1 = FROM
                              \ Addr2 = TO
                              \ N1    = Number of Bytes
1- 0 DO          \ --- Addr-from Addr-To )
OVER I + TC@     \ --- Addr-from Addr-To UC1 )
OVER I + TC!
LOOP 2DROP
;

: TFILL                       \  ( Addr1 N1 C1 ---       )
                              \  Addr1 = FROM
                              \  N1    = Number of Bytes 0001 - FFFF
                              \  C1    = Byte-to-be-filled

SWAP 1- ROT DUP    \  --- C1 N1' Addr1 Addr1 )
ROT + SWAP
DO DUP I TC! LOOP DROP
;

: TDUMP                       \ ( Addr1  ---    )
                              \ prints the content of Addr1 and the
                              \ following 15 Bytes in hex

5 SPACES
F 0 DO
```

```
161   I DIGIT EMIT 2
162   I 03 AND 03 = IF 1+ THEN SPACES
163   LOOP
164
165   DUP
166
167   BEGIN
168     CR FFF0 AND DUP NH.
169     F 0 DO                        \ ( Addr1 Addr2 ---     )
170
171     DUP I + HOPP 2DUP U< LNOT
172     IF DROP TC@ CH. ELSE 2DROP 3 SPACES THEN
173
174     I 03 AND 03 = IF SPACE THEN
175     LOOP
176
177   KEY 20 =
178   IF
179   \ DUP 230 =                     \ upper end of RAM
180   \ IF DROP 1 ELSE
181
182   10 + 0
183   \ THEN
184
185   ELSE DROP 1 THEN
186
187   UNTIL
188
189   DROP
190
191   CR ;
192
193
```

Wave Engine (7)

Hannes Teich

Nachdem in der letzten Folge die erste Ausbaustufe der neuen Version der Wave Engine beschrieben und geliefert worden ist, geht es nun mit der aufgebohrten zweiten weiter. Ein Tipp vorab: Wer die begleitenden Dateien zum Selber-Spielen nicht finden kann, der möge auf meiner Homepage (siehe am Schluss) nachsehen, da sind sie allemal vorhanden.

Mehrstimmigkeit

Die kürzlich beschriebene Version 3-01 (Wave-Engine-Ver3-01.zip) zeigt, wie berechnete Sinustöne in einer abspielbaren Wave-Datei abgelegt werden. Die Erzeugung der Sinuskurven war recht simpel, braucht aber in der vorliegenden Version 3-02 (Wave-Engine-Ver3-02.zip) der Mehrstimmigkeit wegen erheblich mehr Aufwand. Nach der Berechnung eines Kurvenwertes muss zur nächsten Kurve weitergegangen werden, bis alle Kurven berücksichtigt wurden, dann erst ist die erste Kurve wieder an der Reihe. Die „gleichzeitigen“ Werte werden in einem Akkumulator aufaddiert und zum Wave-Generator geschickt, der die Wave-Datei zusammensetzt.

Es geht hier freilich noch längst nicht ums Musizieren, sondern erst mal ums Prinzip. Der Vollausbau wird wieder 4 „Manuale“ haben, jedes mit 6 Stimmen ausgestattet, jede Stimme mit 15 Obertönen versehen. Hier und heute gibt es 2 Manuale mit je 2 Stimmen, und jede Stimme hat einen Oberton (einen 3. Teilton, eine Duodezime höher als der Grundton). Das zeigt das Prinzip und bleibt übersichtlich. Stereo und Rechteck der Version 3-01 sind diesmal weggelassen worden.

Geheimnisvolles Rauschen

Es war vereinbart worden, diesen Beitrag im August zu liefern. Wenn ich's recht bedenke, endet bei Niederschrift dieser Zeilen der August in zwei Stunden, weshalb diese Version 3-02 nun so, wie sie ist, an die Vierte Dimension abgeschickt wird. Was mich stört, aber nicht mehr korrigiert werden kann, ist ein geheimnisvolles störendes Rauschen beim Abspielen der vier 4-stimmigen Akkorde, die als Beispiel gegeben werden. Vielleicht hat ja jemand Lust, selber nach der Ursache zu forschen? Ich gelobe jedenfalls, mit Version 3-03 das Rauschen verschwinden zu lassen. Es rauschte ja früher auch nicht, wie man sich auf meiner Homepage überzeugen kann. Abgedruckt wird die Datei singen-1.fs (Sinus-Generator), die gegenüber singen.fs deutlich verändert worden ist.

Referenzen

In diesen VD-Heften wurde bisher über die Wave Engine berichtet:
2/2011 Top-One-Partitur, 3/2011, 1/2012, 2/2012, 4/2012, 1/2013, 2/2013.
Beachten Sie bitte den Dateibereich der Website der Forth-Gesellschaft unter `http://www.forth-ev.de/filemgmt/viewcat.php?cid=54`
oder alternativ `http://tinyurl.com/WEpage`
sowie die Website des Autors unter `http://www.stocket.de/WE`

Listing

```
1   \ ====== 1 ======= 2 ======= 3 ======= 4 ======= 5 ======= 6 ======= 7 ==|
2   \ singen-1.fs - last edit: 31.08.2013 19:00 -jgt
3     cr ." included: singen-1.fs
4   \
5   \ ------------------------------------------------------------------------
6   \
7   \ Doppelbyte-Speicherzugriffe
8   \
9   \ w!   ( n a --)                          \ in Gforth vordefiniert
10  \ uw@  ( a -- u)    ( unsigned)           \ in Gforth vordefiniert
11  \ w@   ( a -- u)    ( unsigned = Fehler)  \ in Gforth vordefiniert
12  : w@'  ( a -- n)    ( signed)             \ Workaraound fuer w@
13                    uw@ dup $8000 and IF $-10000 or THEN ;
14  : wn+  ( n a --)  dup uw@ rot + swap w! ;   \ n addieren
15  : w1+  ( a --)    dup uw@ 1+ swap w! ;      \ 1 addieren
16  : w1-  ( a --)    dup uw@ 1- swap w! ;      \ 1 subtrahieren
17  : w,   ( --)      here 2 allot w! ;         \ 1 Doppelbyte ablegen
18  \
19  \ ------------------------------------------------------------------------
```

```
\ | 0xxx xxxx xxxx xxxx |   Tondauer (1 bis 28800 = 10 Takte)
\ | 10mm sssx xxxx xxxx |   Pitch (max 109) (mm/sss = Manual/Stimme)
\ | 10mm sss0 0000 0000 |   Pause
\ | 10mm sss1 1111 1111 |   Fortklingen
        2  constant   #manuals      \ Anzahl Manuale
        2  constant   #voices       \ Anzahl Stimmen pro Manual
        2  constant   #partials     \ Anzahl Teiltoene pro Stimme
  .99983e  fconstant  damp          \ Daempfung (Abklingen)

        0  value      manu#         \ Manual #
        0  value      lvox#         \ Stimme # lokal
        0  value      gvox#         \ Stimme # global
        0  value      lpart#        \ Teilton # lokal
        0  value      gpart#        \ Teilton # global
        0  value      durat         \ Tondauerzaehler
        0  value      habit         \ Anzahl Stimmen (not used)
           fvariable  accu          \ Sinuswerte aufaddieren
\ -------------------------------------------------------------------------
\ Tabelle der Float-Werte, jedem Teilton zugeordnet.
\ 2 Manuale, 4 Stimmen, 8 Teiltoene (kuenftig 4 Manuale, 24 Stimmen).
\ je Teilton (floating) Phase (p), Phasenschritt (s), Huellkurve (e).
\
create gtable  #manuals #voices #partials 24 * * * allot
\
\     ------------------------------------------------
\     |                    manual-A                  |
\     |         voice1        |         voice2        |   p = phase
\     |   part1   |   part2   |   part1   |   part2   |   s = step
\     | p | s | e | p | s | e | p | s | e | p | s | e |   e = envelope
\     ------------------------------------------------
\       0   8  16  24  32  40  48  56  64  72  80  88 (96)
\     ------------------------------------------------
\     |                    manual-B                  |
\     |         voice1        |         voice2        |
\     |   part1   |   part2   |   part1   |   part2   |
\     | p | s | e | p | s | e | p | s | e | p | s | e |
\     ------------------------------------------------
\      96 104 112 120 128 136 144 152 160 168 176 184 (192)
\ -------------------------------------------------------------------------
\ Sinus-Stuetzpunkte berechnen
\ Erster Aufruf: Timer gesetzt, Phase=0, Phasenschritt gesetzt.
\ Der Sinuswert (max. plus/minus 1) mal "volume" wird aufaddiert.
\ Die Phase wird um den Phasenschritt erhoeht, der Timer erniedrigt.
\
: generate  ( --)
    gpart# 24 * gtable + dup f@ fsin            \ Sinus aus Phase
                       dup 16 + f@ f*           \ mal Huellkurve
                   accu dup f@ f+ f!            \ in Akku aufaddieren
                       dup f@                   \ Phase lesen
                       dup  8 + f@ f+           \ plus Phasenschritt
                       dup f!                   \ Phase schreiben
                   16 + dup f@  damp f* f! ;    \ Huellkurve daempfen

\ -------------------------------------------------------------------------
: >voice  ( voice manual -- gvoice)             \ Stimmen-Nummer
    #voices * + ;
: >partial  ( partial voice manual -- gpart)    \ Teilton-Nummer
    >voice  #partials * + ;
: >mpart  ( partial manual -- mpart)            \ Teilton eines Manuals
    #partials * + ;
```

```
\ Schleifen fuer Manuale, Stimmen und Teilstimmen
\ Fuer jede Teilstimme wird ein Kurvenpunkt berechnet.
\
: loops  ( --)
    durat 0
    ?DO
        0e accu f!                                  \ Akku leeren
        #manuals 0
        ?DO  i                          to manu#    \ Manual-Nummer
            #voices 0
            ?DO  i                      to lvox#    \ Stimmen-Nummer
                i j           >voice    to gvox#    \ dito fortlaufend
                #partials 0
                ?DO  i                  to lpart#   \ Teilton-Nummer
                    i j k    >partial   to gpart#   \ dito fortlaufend
                    generate                        \ Teilton bearbeiten
                LOOP
            LOOP
        LOOP
        accu f@  f>s w>buf              \ in die Wave-Datei
    LOOP ;
\ ------------------------------------------------------------------------
\ Stimmen laden (Phase, Step, Envelope fuer Grund- und 2. Oberton)
: >A1  ( u --)           0e gtable       f!     0e gtable  24 + f!
             fadr12 f@ fdup gtable   8 + f!  3e f* gtable  32 + f!
                       20e gtable  16 + f!    20e gtable  40 + f! ;
: >A2  ( u --)           0e gtable  48 + f!     0e gtable  72 + f!
             fadr12 f@ fdup gtable  56 + f!  3e f* gtable  80 + f!
                       20e gtable  64 + f!    20e gtable  88 + f! ;
: >B1  ( u --)           0e gtable  96 + f!     0e gtable 120 + f!
             fadr12 f@ fdup gtable 104 + f!  3e f* gtable 128 + f!
                       20e gtable 112 + f!    20e gtable 136 + f! ;
: >B2  ( u --)           0e gtable 144 + f!     0e gtable 168 + f!
             fadr12 f@ fdup gtable 152 + f!  3e f* gtable 176 + f!
                       20e gtable 160 + f!    20e gtable 184 + f! ;
\ Mini-Partitur: vier 4-stimmige Akkorde
create tune
\   Frames        Anzahl    A1      A2      A3      A4
    30000 w,      4 w,      49 w,   44 w,   41 w,   37 w,
    30000 w,      4 w,      49 w,   46 w,   42 w,   30 w,
    30000 w,      4 w,      48 w,   44 w,   39 w,   32 w,
    60000 w,      4 w,      49 w,   44 w,   41 w,   37 w,
4 constant tunelen ( Zeilen)
\ Partiturschleife: Akkord fuer Akkord
\ Phase=0, Timer und Tonhoehen laden, Generator starten
: waves  ( --)
    tunelen 0                       \ Partiturlaenge
    ?DO
        tune i 12 * +               \ 10 Bytes (2 Tondauer, 4*2 Tonhoehe)
        dup      uw@  to durat      \ Tondauer (fuer alle Teiltoene)
        dup  2 + uw@  to habit      \ 4 = 4 Werte folgen (not used)
        dup  4 + uw@  >A1           \ Manual A. Stimme 1
        dup  6 + uw@  >A2           \ Manual A, Stimme 2
        dup  8 + uw@  >B1           \ Manual B, Stimme 1
            10 + uw@  >B2           \ Manual B, Stimme 2
        loops
    LOOP ;
\ =======[ Ende von singen-1 ]=============================================
```

Forth–Gruppen regional

Mannheim **Thomas Prinz**
Tel.: (0 62 71) – 28 30 (p)
Ewald Rieger
Tel.: (0 62 39) – 92 01 85 (p)
Treffen: jeden 1. Dienstag im Monat
Vereinslokal Segelverein Mannheim e.V. Flugplatz Mannheim–Neuostheim

München **Bernd Paysan**
Tel.: (0 89) – 41 15 46 53 (p)
`bernd.paysan@gmx.de`
Treffen: Jeden 4. Donnerstag im Monat um 19:00 in der Pizzeria La Capannina, Weitlstr. 142, 80995 München (Feldmochinger Anger).

Hamburg Küstenforth
Klaus Schleisiek
Tel.: (0 40) – 37 50 08 03 (g)
`kschleisiek@send.de`
Treffen 1 Mal im Quartal
Ort und Zeit nach Vereinbarung (bitte erfragen)

Mainz Rolf Lauer möchte im Raum Frankfurt, Mainz, Bad Kreuznach eine lokale Gruppe einrichten.
Mail an rowila@t-online.de

Gruppengründungen, Kontakte

Hier könnte Ihre Adresse oder Ihre Rufnummer stehen — wenn Sie eine Forthgruppe gründen wollen.

μP–Controller Verleih

Carsten Strotmann
`microcontrollerverleih@forth-ev.de`
`mcv@forth-ev.de`

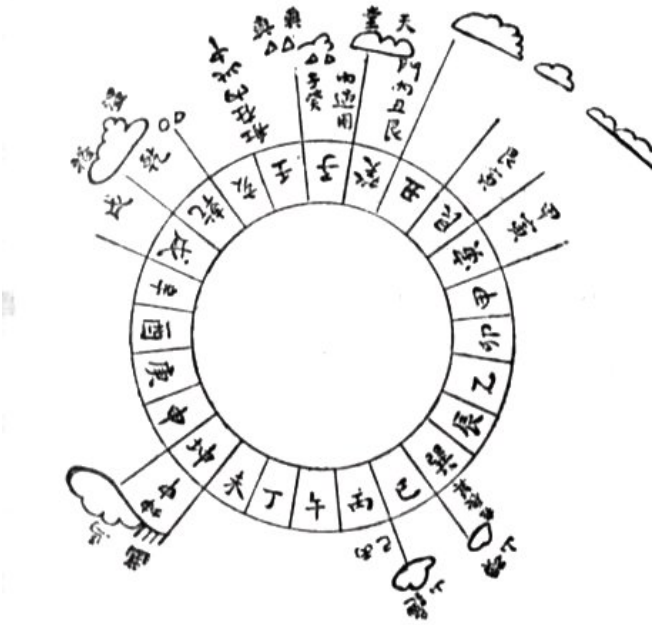
Spezielle Fachgebiete

FORTHchips (FRP 1600, RTX, Novix)	**Klaus Schleisiek–Kern** Tel.: **(0 40) – 37 50 08 03** (g)
KI, Object Oriented Forth, Sicherheitskritische Systeme	**Ulrich Hoffmann** Tel.: **(0 43 51) – 71 22 17** (p) Fax: **– 71 22 16**
Forth–Vertrieb volksFORTH ultraFORTH RTX / FG / Super8 KK–FORTH	Ingenieurbüro **Klaus Kohl–Schöpe** Tel.: **(0 82 66)–36 09 862** (p)

Termine

Donnerstags ab 20:00 Uhr
Forth–Chat IRC `#forth-ev`

25.–27. September 2013: Forth200x meeting
27.–29. September 2013: EuroForth 2013 conference
09.–10. November 2013: OpenRheinRuhr, Oberhausen



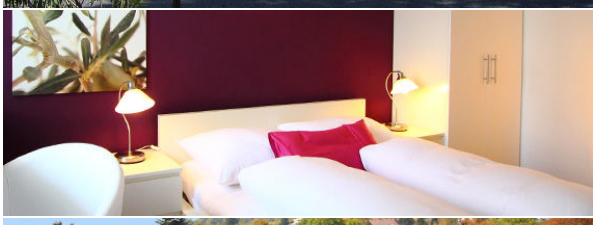
Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen? Schreiben Sie einfach der VD — oder rufen Sie an — oder schicken Sie uns eine E–Mail!

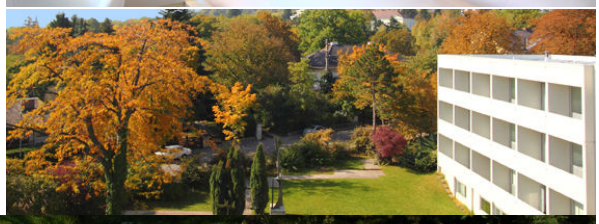
Hinweise zu den Angaben nach den Telefonnummern:
Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich
Die Adressen des Büros der Forth–Gesellschaft e.V. und der VD finden Sie im Impressum des Heftes.

Einladung zur

Forth–Tagung 2014 vom 27. bis 30. März in Bad Vöslau

Unterbringung und Tagung im College Garden Hotel in der Johann–Strauß–Straße 2, 2540 Bad Vöslau, Österreich.





Anreise

Bad Vöslau liegt südlich von Wien und ist mit dem Auto über die A2 erreichbar. Das Hotel bietet einen Taxiservice vom Bahnhof, oder auch vom Flughafen.

Anmeldung

Auf `http://tagung.forth-ev.de` finden sich noch viele weitere Informationen, das aktuellste Programm sowie die elektronische Anmeldung.

Programm

Donnerstag

ab 13:00	Frühankommer(–Workshops)

Freitag

vormittags	Frühankommer(–Workshops)
nachmittags	Begin der Tagung, Vorträge und Workshops

Samstag

vormittags	Vorträge und Workshops
nachmittags	Exkursion

Sonntag

09:00	Mitgliederversammlung
13:00	Ende der Tagung

Quellen: wikimedia.org, college-garden-hotels.com