THAT Analog Computer Book

Collection of Examples, useful Hints and Electronic Circuits

by

Michael Koch, astroelectronic@t-online.de

Version from February 14, 2024

Content

1 Analog Computers

1.1 Useful Links for THAT

THAT Website: https://the-analog-thing.org/

Wiki: https://the-analog-thing.org/wiki/Main_Page

First steps book: https://the-analog-thing.org/THAT_First_Steps.pdf

Schematic diagrams of THAT: https://the-analog-thing.org/wiki/The_Analog_Thing#Schematics

Files on Github: https://github.com/anabrid/the-analog-thing/tree/main/ANATHING_OSH

Anabrid Model 1 schematics on Github: https://github.com/anabrid/Model-1/tree/main/hardware

THAT Newsletter archive: https://the-analog-thing.org/newsletter/

1.2 Useful Links for Analog Computing

Bernd Ulmann's book "Analog and Hybrid Computer Programming": https://www.amazon.de/Analog-Computer.../dp/3110662078

Analog Computer Museum: https://analogmuseum.org/

Analog computer museum's Online Library: https://analogmuseum.org/english/library.html

EAI Handbook of Analog Computation (from 1971): https://analogmuseum.org/library/eai_handbook.pdf

This is an older version of the above book, from 1967: http://www.bitsavers.org/pdf/eai/EAI_Handbook_of_Analog_Computation_1967.pdf

Leybold-Heraeus, Der Analogrechner: https://analogmuseum.org/library/der_analogrechner.pdf

Systron Donner, Handbook of Analog Computation: https://analogmuseum.org/library/handbook_of_analog_computation.pdf

This book: http://www.astro-electronic.de/THAT_Analog_Computer_Book.pdf

James M. Fiore: Operational Amplifiers and Linear Integrated Circuits - Theory and Application
https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Operational_Amplifiers_and_Linear_Integrated_Circuits_-_Theory_and_Application_(Fiore)

Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

Examples for analog computing: https://www.analogmuseum.org/english/examples/

List of application notes for analog computers: https://analogparadigm.com/documentation.html

All issues from Analog Devices "Analog Dialogue": https://www.analog.com/en/analog-dialogue/archives.html

Volume 1, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-1/number-1/articles/volume1-number1.pdf

Page 6: Operational integrators

Page 9: Comparison of capacitor specifications

Volume 1, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-1/number-2/articles/volume1-number2.pdf

Page 14: Overload clamp circuit

Volume 2, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-2/number-1/articles/volume2-number1.pdf

Page 12: Ideal rectifier

Volume 5, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-5/number-2/articles/volume5-number2.pdf

Page 15: Amplitudes without delay, measuring sine wave amplitudes without filtering

Volume 5, Number 5: https://www.analog.com/media/en/analog-dialogue/volume-5/number-5/articles/volume5-number5.pdf

Page 14: Sine oscillator

Volume 8, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-8/number-1/articles/volume8-number1.pdf

Page 13: RMS Measurement, Crest Factor

Volume 8, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-8/number-2/articles/volume8-number2.pdf

Page 23: Errata for Nonlinear Circuits Handbook

Volume 11, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-11/number-1/articles/volume11-number1.pdf

Page 6: AD534 Multiplier

Page 18: Errata for Nonlinear Circuits Handbook

Volume 12, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-12/number-3/articles/volume12-number3.pdf

Page 16: Vector Generators DTM1716 and DTM1717

Volume 13, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-13/number-2/articles/volume13-number2.pdf

Page 4: A/D converter with seriel (cascade) Gray code

Page 11: High resolution differential temperature measurement with AD590

Page 18: Four quadrant divider

Volume 14, Number 2:

Page 18: Some notes about the high resolution differential temperature measurement from volume 13, number 2.

Volume 16, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-16/number-2/articles/volume16-number2.pdf

Page 5: State variable filter

Volume 16, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-16/number-3/articles/volume16-number3.pdf

Page 20: AD539 Dual analog multiplier

Volume 17, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-17/number-1/articles/volume17-number1.pdf

Page 8: Shielding and Guarding

Volume 17, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-17/number-2/articles/volume17-number2.pdf

Page 11: Guarding

Page 12: Digital FIR Filters

Volume 18, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-18/number-1/articles/volume18-number1.pdf

Page 3: DC Amplifier Noise

Page 11: AD637 RMS to DC Converter

Page 14: Fast, simple Approximation of Functions

Volume 18, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-18/number-2/articles/volume18-number2.pdf

Page 16: AD630 Balanced Modulator / Demodulator

Volume 18, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-18/number-3/articles/volume18-number3.pdf

Page 12: AD639 Trigonometric Function Generator

Volume 19, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-19/number-1/articles/volume19-number1.pdf

Page 3: AD538 Analog Multifunction Chip

Volume 20, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-20/number-2/articles/volume20-number2.pdf

Page 19: LVDT (Linear Variable Differential Transformer)

Volume 21, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-21/number-1/articles/volume21-number1.pdf

Page 10: 2S81 Resolver to Digital Converter

Page 12: AD588 Precision Voltage Reference

Volume 22, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-22/number-2/articles/volume22-number2.pdf

Page 12: AD840 Series

Page 16: AD9300 Video Multiplexer, Chroma Keying

Volume 23, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-23/number-1/articles/volume23-number1.pdf

Page 7: AD689 8.192V Reference

Page 8: AD834 4-Quadrant Multiplier

Volume 23, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-23/number-3/articles/volume23-number3.pdf

Page 3: AD640 Logarithmic Amplifier

Page 7: Grounding

Page 15: AD598 LVDT Conditioner

Page 16: 6S04 Synchro Simulator and Tester

Volume 24, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-24/number-1/articles/volume24-number1.pdf

Page 12: some infos about Synchros and Resolvers

Volume 24, Number 3: https://www.analog.com/media/en/analog-dialogue/volume-24/number-3/articles/volume24-number3.pdf

Page 20: AD734 Multiplier

Page 20: AD688 +-10V Reference

Volume 25, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-25/number-1/articles/volume25-number1.pdf

Page 10: AD9950 Direct Digital Synthesis

Page 23: AD633 Multiplier

Volume 26, Number 1: https://www.analog.com/media/en/analog-dialogue/volume-26/number-1/articles/volume26-number1.pdf

Page 10: AD811 Video Operational Amplifier

Volume 26, Number 2: https://www.analog.com/media/en/analog-dialogue/volume-26/number-2/articles/volume26-number2.pdf

Page 13: OP Amps in Line Driver and Receiver Circuits, Part 1

Volume 27, Number 1: **https://www.analog.com/media/en/analog-dialogue/volume-27/number-1/articles/volume27-number1.pdf**

Page 14: OP Amps in Line Driver and Receiver Circuits, Part 2

Volume 28, Number 1: **https://www.analog.com/media/en/analog-dialogue/volume-28/number-1/articles/volume28-number1.pdf**

Page 13: Voltage References

Volume 28, Number 3: **https://www.analog.com/media/en/analog-dialogue/volume-28/number-3/articles/volume28-number3.pdf**

Page 6: Clamping

Volume 29, Number 1: **https://www.analog.com/media/en/analog-dialogue/volume-29/number-1/articles/volume29-number1.pdf**

Page 8: Digitally controllable variable Resistors

Page 13: Clamp Amplifiers

Volume 29, Number 3: **https://www.analog.com/media/en/analog-dialogue/volume-29/number-3/articles/volume29-number3.pdf**

Page 15: AD7805

Volume 30, Number 3: **https://www.analog.com/media/en/analog-dialogue/volume-30/number-3/articles/volume30-number3.pdf**

Page 20: Current Feedback Amplifiers, Part 1

Volume 30, Number 4: **https://www.analog.com/media/en/analog-dialogue/volume-30/number-4/articles/volume30-number4.pdf**

Page 20: Current Feedback Amplifiers, Part 2

Volume 31, Number 2: **https://www.analog.com/media/en/analog-dialogue/volume-31/number-2/articles/volume31-number2.pdf**

Page 19: OP Amps driving capacitive Loads

....

Volume 42, Number 4: **https://www.analog.com/media/en/analog-dialogue/volume-42/number-4/articles/volume42-number4.pdf**

Page 8: Considering Multipliers, Part 1

1.3 Other useful Links

My FFmpeg Book: http://www.astro-electronic.de/FFmpeg_Book.pdf

My Teensy Book: http://www.astro-electronic.de/Teensy_Book.pdf

My C# Programming Book (in german language): http://www.astro-electronic.de/CSharp_Buch.pdf

1.4 Useful Books

Analog Devices: Nonlinear Circuits Handbook, Second Edition January 1976, ISBN: 0-916550-01-X

Analog Devices: The Best of Analog Dialogue, ISBN 0-9-916550-10-9

Analog Devices: High Speed Design Techniques, ISBN 0-916550-17-6

Tietze, U. and Schenk, Ch.: Halbleiter-Schaltungstechnik, 9. Auflage, ISBN 3-540-19475-4

Tietze, U. and Schenk, Ch.: Halbleiter-Schaltungstechnik, 11. Auflage, ISBN 3-540-64192-0

Pease, Robert A.: Troubleshooting Analog Circuits, ISBN 0-7506-9499-8

Horowitz, Paul and Hill, Winfield: The Art of Electronics, Second Edition, ISBN 0-521-37095-7

Horowitz, Paul and Hill, Winfield: The Art of Electronics, Third Edition, ISBN 978-0-521-80926-9

Williams, Arthur and Taylor, Fred: Electronic Filter Design Handbook, ISBN 0-07-147171-5

1.5 Data Sheets for ICs

Analog Devices:

Multipliers and Dividers: https://www.analog.com/en/parametricsearch/11106#/

Logarithmic Transimpedance Amplifiers: https://www.analog.com/en/parametricsearch/10772#/

AD534: [(X1 − X 2) * (Y1 − Y 2) / 10 V] + Z 2 https://www.analog.com/media/en/technical-documentation/data-sheets/AD534.pdf

AD538: $V_O = V_Y * (V_Z / V_X)^M$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD538.pdf

AD633: W = (X1 - X2) * (Y1 - Y2) / 10V + Z https://www.analog.com/media/en/technical-documentation/data-sheets/AD633.pdf

AD639: Sin, Cos, Tan... https://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/AD639.pdf

AD734: This multiplier is better than AD633, but very expensive. https://www.analog.com/media/en/technical-documentation/data-sheets/AD734.pdf

AD818: Paired transistors for log / antilog Note: The same type number is also used for a video amplifier chip, which is a completely different thing! https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf

BD2244G-M This current limiter is used at the USB power input port, limiting the current to 590mA.

Data sheet: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/power_switch/bd2244g-m-e.pdf

Texas Instruments / Burr-Brown:

LOG112, LOG2112, Precision Logarithmic and Log Ratio Amplifiers: https://www.ti.com/lit/ds/symlink/log112.pdf

Op-Amps:

Type	Supply Voltage	Offset Voltage		Output Current	Slew Rate	Package	Link to Datasheet, Notes
		Typical	Maximum				
AD711	+-4.5 V to +-18 V	+-0.3 mV	+-2 mV	25 mA	16 V/µs	DIP8, SO8	https://www.analog.com/media/en/technical-documentation/data-sheets/ad711.pdf
AD8531, AD8532, AD8534	2.7 V to 6.0 V	no spec	+-25mV	250 mA	5 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/AD8531_8532_8534.pdf Rail-to-Rail I/O
AD8541, AD8542, AD8544	2.7 V to 5.5 V	+-1 mV	+-6 mV	15 mA	0.75 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/AD8541_8542_8544.pdf Rail-to-Rail I/O https://www.reichelt.de/operationsverstaerker-2-fach-0-92-v-s-1-mhz-so-8-ad-8542-arz-p185453.html
AD8601A, AD8602A, AD8604A	2.7 V to 5.5 V	+-0.08 mV	+-0.5 mV	30 mA	5 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/ad8601_8602_8604.pdf Rail-to-Rail I/O https://www.reichelt.de/index.html?ACTION=446&LA=446&nbc=1&q=ad8602
AD8601D, AD8602D, AD8604D		+-1.3 mV	+-6 mV				
AD8628, AD8629, AD8630	2.7 V to 5.0 V	+-0.001 mV	+-0.005 mV	30 mA	1 V/µs	SO, TSSOP	https://www.analog.com/media/en/technical-documentation/data-sheets/ad8628-8629-8630.pdf Rail-to-Rail I/O, Ultra-low offset voltage https://www.reichelt.de/operationsverstaerker-2-fach-1-v-s-2-5-mhz-msop-8-ad-8629-armz-p185470.html
LTC1051, LTC1053	4.75 V to 16V	+-0.0005 mV	+-0.005 mV		4 V/µs	DIP	https://www.analog.com/media/en/technical-documentation/data-sheets/10513fa.pdf Ultra-low offset voltage, zero drift
OPA277P, OPA277U	+-2 V to +-18 V	+-0.01 mV	+-0.02 mV	35 mA	0.8 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa277.pdf Ultra-low offset voltage
OPA2277P, OPA2277U		+-0.01 mV	+-0.025 mV				
OPAx277PA, OPAx277UA		+-0.02 mV	+-0.05 mV				
OPAx354	2.5 V to 5.5 V	+-2 mV	+-8 mV	100 mA	150 V/µs		https://www.ti.com/lit/ds/symlink/opa354.pdf Rail-to-Rail I/O
OPA445	+-10 V to +-45 V	+-1.5 mV	+-5 mV	26 mA	15 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa445.pdf
OPA454	+-5 V to +-50 V	+-0.2 mV	+-4 mV	50 mA	13 V/µs	SO	https://www.ti.com/lit/ds/symlink/opa454.pdf
OPA551, OPA552	+-4 V to +-30 V	+-1 mV	+-3 mV	200 mA	15 V/µs, 24 V/µs	DIP, SO	https://www.ti.com/lit/ds/symlink/opa551.pdf
TL064C	+-5 V to +-15 V	+-3 mV	+-15 mV	no spec	3.5 V/µs	DIP, SO,	https://www.ti.com/lit/ds/symlink/tl064.pdf

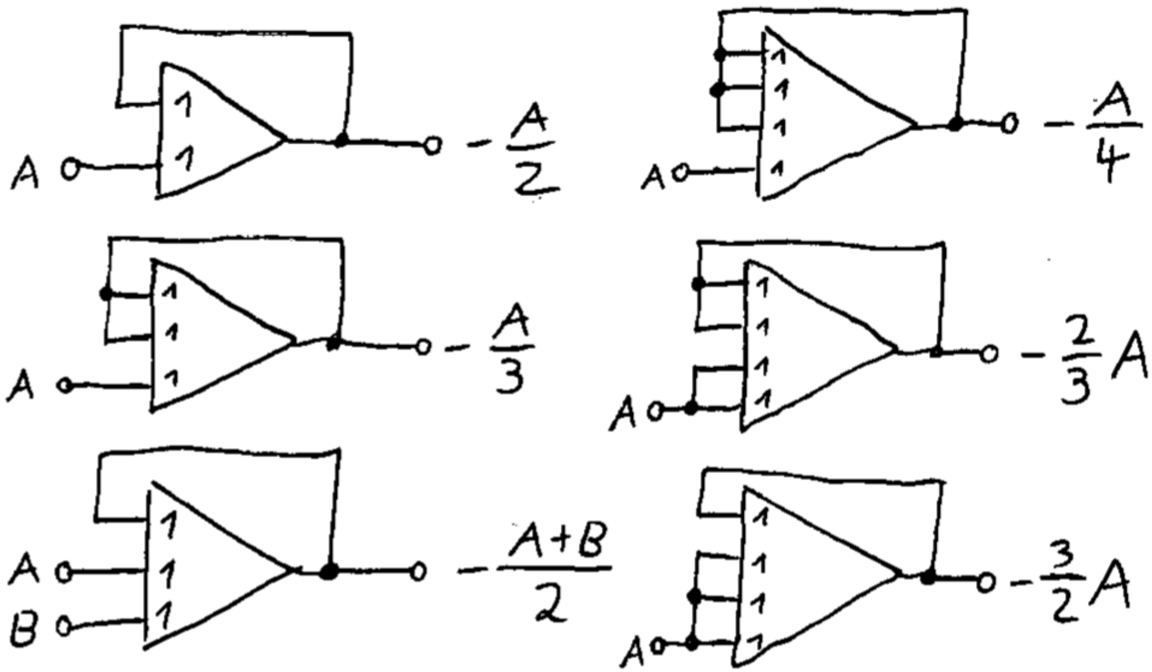
TL064AC, TL064I		+-3 mV	+-6 mV			TSSOP	
TL064BC		+-2 mV	+-3 mV				
TL071C, TL072C, TL074C	+-2.25 V to +-20V	+-3 mV	+-10 mV	26 mA	20 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tl074h.pdf TL074H is used in THAT Historically the TL07x and TL08x had different specs, but nowadays they are almost the same. TL07x had smaller noise and offset voltage.
TL074AC, TL074I		+-3 mV	+-6 mV				
TL074BC		+-2 mV	+-3 mV				
TL074M		+-3 mV	+-9 mV				
TL074H		+-1 mV	+-4 mV				
TL081C, TL082C, TL084C	+-2.25 V to +-20V	+-3 mV	+-15 mV	26 mA	20 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tl084h.pdf Historically the TL07x and TL08x had different specs, but nowadays they are almost the same. TL07x had smaller noise and offset voltage.
TL084AC, TL084I		+-3 mV	+-6 mV				
TL084BC		+-2 mV	+-3 mV				
TL084H		+-1 mV	+-4 mV				
TLC271	3V to 16V					DIP	https://www.ti.com/lit/ds/symlink/tlc271.pdf old design, do not use
TLC272	3V to 16V	+-1.1 mV	+-10 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc272.pdf
TLC274	3V to 16V	+-1.1 mV	+-10 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc274.pdf
TLC277	3V to 16V	+-0.2 mV	+-0.5 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc272.pdf
TLC279	3V to 16V	+-0.32 mV	+-0.9 mV		3.6 V/µs	DIP	https://www.ti.com/lit/ds/symlink/tlc274.pdf
TLC2272, TLC2274	+-2.2 V to +-8 V	+-0.3 mV	+-2.5 mV		3.6 V/µs	DIP, SO, TSSOP	https://www.ti.com/lit/ds/symlink/tlc2272.pdf Rail-to-Rail Output (but not Input)
TLC2272A, TLC2274A		+-0.3 mV	+-0.95 mV				

Power OP-Amps:

Type	Supply Voltage	Offset Voltage		Output Current	Slew Rate	Package	Link to Datasheet or Demoboard, Notes
		Typical	Maximum				
OPA541	+-5 V to +-40 V	+-2 mV	+- 10 mV	5 A cont, 10 A peak	10 V/µs	11-pin TO-220	https://www.ti.com/lit/ds/symlink/opa541.pdf https://de.aliexpress.com/item/1005001523037649.html https://de.aliexpress.com/item/1005005046010895.html
OPA548	+-4 V to +-30 V	+-2 mV	+- 10 mV	3 A cont, 5 A peak	10 V/µs	7-pin TO-220	https://www.ti.com/lit/ds/symlink/opa548.pdf https://de.aliexpress.com/item/1005005046010895.html
OPA549	+-4 V to +-30 V	+-1 mV	+-5 mV	8 A cont, 10 A peak	9 V/µs	11-pin power	https://www.ti.com/lit/ds/symlink/opa549.pdf https://de.aliexpress.com/item/1005005046010895.html

2 Components

2.1 Summer



Please note that the output levels are different in open amplifiers and summers!

See also: https://the-analog-thing.org/wiki/Summer

2.2 Open Amplifier

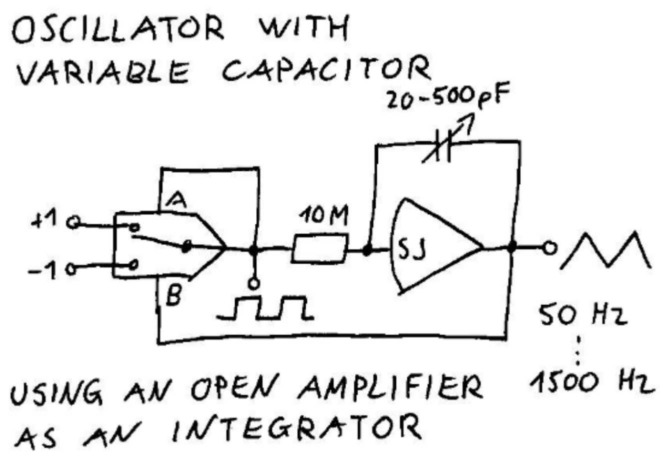
The open amplifier adjusts its output so that the sum of the inputs becomes zero. This does only work if the sign isn't changed in the feedback loop.

An open amplifier can also be used as an integrator. The advantage is that this integrator keeps its value when THAT is in stop mode. The drawback is that you can't set an initial condition.

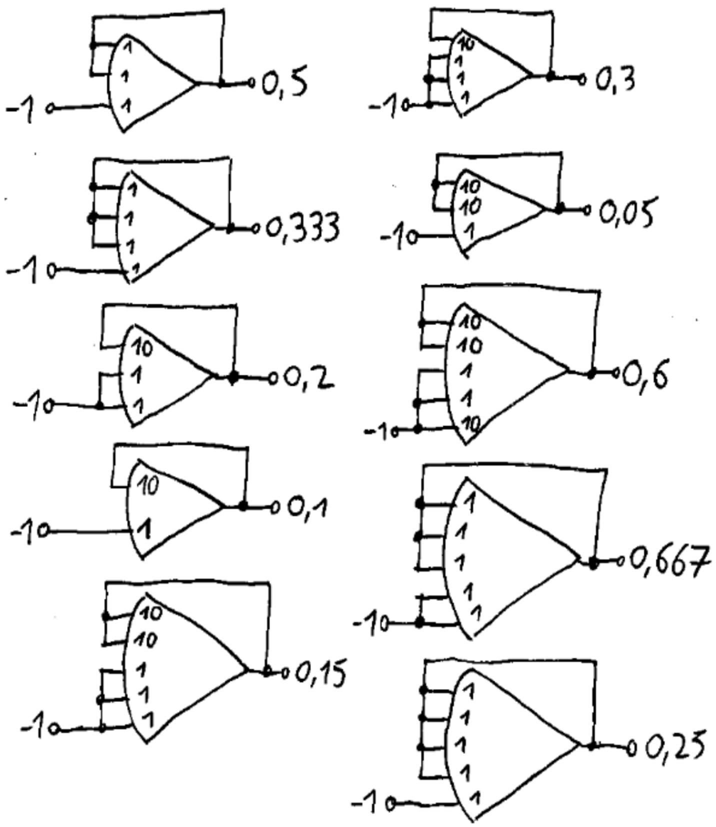
This can also be used if you need an integrator with a longer time scale factor.
1µF * 1MΩ = 1s   10µF * 1MΩ = 10s   10µF * 10MΩ = 100s   10µF * 100MΩ = 1000s
100µF * 100MΩ = 10000s   10µF * 1GΩ = 10000s   3.6µF * 1GΩ = 3600s = 1h   100µF * 36MΩ = 3600s = 1h

100 MΩ resistors are available from Farnell: https://de.farnell.com/te-connectivity/rgp0207chj100m/dicksch-widerstand-100m-5-250mw/dp/2805251

1 GΩ resistors are also available from Farnell: https://de.farnell.com/te-connectivity/rgp0207chk1g0/dickschichtwiderstand-1g-10-250mw/dp/2805261



An open amplifier can also be used for making precision constants, without having to adjust coefficients:

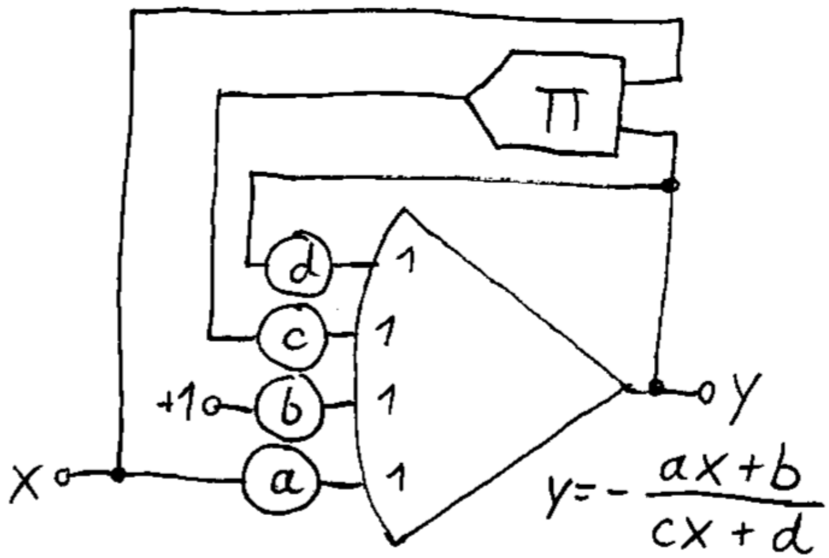


For negative constants, simply replace the -1 input by +1 input.

Please note that the output levels are different in open amplifiers and summers!

See also https://the-analog-thing.org/wiki/Summer

Function y = -(ax + b) / (cx + d)



$$ax + b + cxy + dy = 0$$

2.3 Operational amplifiers with vacuum tubes

https://www.allaboutcircuits.com/textbook/semiconductors/chpt-8/operational-amplifier-models/

https://www.rfcafe.com/references/electronics-world/operational-amplifier-electronics-world-july-1963.htm

12AX7 = ECC83 https://www.tube-town.net/ttstore/roehren/vorstufenroehren/12ax7-ecc83-5751/?language=de

2.4 Integrator

The integration time scale factor k_0 in THAT is 1ms, or 100ms in SLOW mode. If you need 10ms time scale factor, you can connect an external 10nF capacitor between OUT and SLOW. The exact value would be 9.89nF, so you can select a capacitor with -1% tolerance.

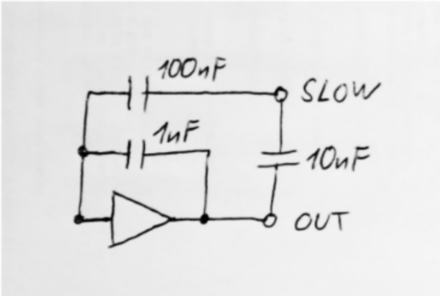
$$OUT(t) = 1 / k_0 * \int IN(t)\,dt \qquad IN(t) = k_0 * dOUT(t) / dt$$

All time constants between 1ms and 100ms can be realized with an external capacitor between SLOW and OUT.

The time constant in ms can be calculated as follows: $k_0 = (100 + 101 * C) / (100 + C)$ where C is the external capacitor in nF.

The external capacitor in nF can be calculated as follows: $C = (100 - 100 * k_0) / (k_0 - 101)$ where k_0 is the time constant in ms.

Time constant k0	External capacitor between SLOW and OUT
1 ms	open
3 ms	2.04 nF
10 ms	9.89 nF
30 ms	40.8 nF
100 ms	short



Integrators with capacitors larger than 100nF can be realized with open amplifiers.

Integrators with time constants larger than 100ms can also be realized by using larger input resistors (10 MΩ, 100 MΩ, 1000 MΩ).

The output of the integrator is inverting. The initial condition is set before the inverter.
That means if you set IC to -1, then the output will initially be at +1.

See also: https://the-analog-thing.org/wiki/Integrator

How fast does the integrator settle to the initial voltage, or with other words: What's the minimum duration of the initial condition?

The time constant is $\tau = 100nF * 10k\Omega = 1ms$ and the voltage in the capacitor follows an exponential function:

$$U = U_0 * e^{(-t/\tau)}$$

As the worst case U_0 can be 2 machine units, if the capacitor must be discharged from +1 MU to -1MU.

IC Duration	Voltage U in capacitor, for U_0 = 2 = Error in machine units
1ms	0.736
2ms	0.271
3ms	0.100
4ms	0.037
5ms	0.013
6ms	0.0050
8ms	0.0007
10ms	0.0001

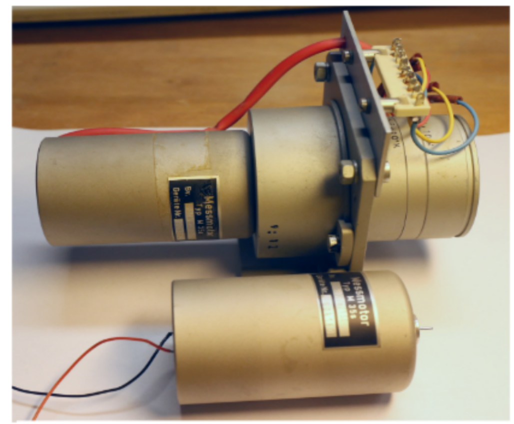
2.5 Electromechanical Integrator

These are motors from an electromechanical integrator, probably from the 1960s or 1970s. One of the motors has a 1:12 gearbox and a precision stereo potentiometer, 2640 Ω and 195 Ω.
Motor without gearbox: "Messmotor Typ M 35s, Bv. 900/a/1St". Motor with gearbox: "Messmotor Typ M 35s, Bv. 1800/a/1St".

The motor current is well below 1mA, so it can directly be connected to a THAT output.

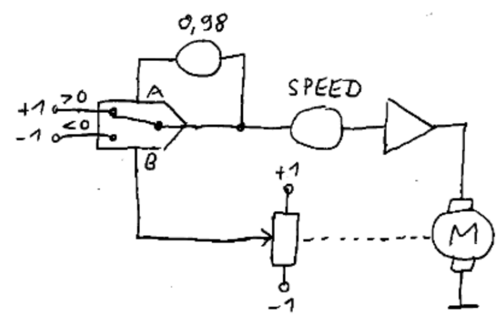
The maximum voltage for the "M 35s Bv. 1800/a/1St" motor is 14V.

The motor begins to run at 0.15V = 0.015 machine units, and the time constant is about 3.25s (potentiometer from end to end, with 10V motor voltage).



My measurement results:

Motor Voltage M 35s 1800/a/1St		Duration t 270° end to end	t * MU This should be a constant	Notes
V	MU			
0.25	0.025	228s	5.7s	Speed is significantly irregular
0.5	0.050	85.2s	4.26s	Speed is slightly irregular
1	0.100	36.8s	3.68s	
1.5	0.150	23.3s	3.49s	
2	0.200	17.2s	3.44s	
2.5	0.250	13.6s	3.37s	
3	0.300	11.3s	3.39s	
4	0.400	8.35s	3.34s	
5	0.500	6.62s	3.31s	
6	0.600	5.49s	3.29s	
7	0.700	4.70s	3.29s	
8	0.800	4.09s	3.27s	
9	0.900	3.63s	3.27s	
10	1.000	3.25s	3.25s	



There are some details about this motor in this document on page 4: https://docplayer.org/204259855-Dehnungsmessanlage-fuer-genaue-leistungsmessungen-an-zapfwellenbetriebenen-landmaschinen-mit-hilfe-von-in-tegratoren.html

The same information is also in this document on page 133: https://mediatum.ub.tum.de/doc/1518280/1518280.pdf

Tafel 1: Auszug aus der Auswahl von Meßmotoren

Meßbereich V	Stromaufnahme bei Nennspannung mA	Meßmotor M 35s Bv	Solldrehzahl bei Meßbereichendwert U/min	Fehler max. % bezogen auf anliegenden Wert bei U_N 100 %	10 %
0 . . . 1.75	1.5	225/a			
0 . . . 3.5	0.9	450/a		ca.	ca.
0 . . . 7	0.5	900/a	240	+ 0.15	— 1.5
0 . . . 14	0.3	1800/a			
0 . . . 24	0.2	3000/a			

[11] Alle Angaben über den integrierenden Impulsgeber wurden einer Druckschrift der Firma Fernsteuergeräte ZACHARIÄ-OELSCH-MAIER in Berlin 47, entnommen

Oelsch, Kurt, Ing. grad., 1000 Berlin 37, Glockenstr. 4, * 15.4.1919
PershGes: Fernsteuergeräte, Zachariä-Oelsch-Meier, 1000 Berlin 47, Jahnstr. 68 - 72
Fernsteuergeräte KG Zachariä-Oelsch-Meier, 6148 Heppenheim (Bergstr), Weiherhausstraße

That means at 10V (1 machine unit) the speed of the "M 35s Bv. 1800/a/1St" motor is 240 / 14 * 10 = 171.4 revolutions per minute, or 2.857 revolutions per second.

After the 1:12 gearbox the speed is 0.238 revolutions per second, or 4.2 seconds for 360° angle or 3.15 seconds for the 270° sector of the potentiometer. It seems my motor is running about 3% too slow.

The company does still exist! https://www.fsg-sensors.de/about

2.6 Differentiator

See chapter 5.14 in Bernd's book "Analog and Hybrid Computer Programming".

Or use an open amplifier with an integrator in the feedback loop.

2.7 Comparator

The comparators in THAT have about 0.022V hysteresis (about 0.002 machine units).

https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic4.pdf

See also https://the-analog-thing.org/wiki/Comparator

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 154ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

2.8 Multiplier

In THAT the AD633JR chip is used. Because this chip is specified for +-15V supply voltage, but used in THAT with +-12V supply, the inputs are divided by 2 and the output is multiplied by 4.

Data sheet: https://www.analog.com/media/en/technical-documentation/data-sheets/AD633.pdf

2.9 AD538 Multiplier

$V_O = V_Y * (V_Z / V_X)^M$ https://www.analog.com/media/en/technical-documentation/data-sheets/AD538.pdf

For the circuit in figure 15 in the data sheet, a special resistor 1090 Ω +3300ppm/K (not exact, see below) is required. 1%/K = 10000 ppm/K

This temperatuire coefficient is in more detail explained in this article (page 3):

https://www.analog.com/media/en/analog-dialogue/volume-19/number-1/articles/volume19-number1.pdf

The required temperature coefficient is 3333 ppm/K or 0.333 %/K. This is simply because the output voltage of the log section is proportional to the absolute temperature. It we assume T=300K (+26.85°C), then the temperature coefficient is 1/300 = 3333ppm.

In my opinion there is an error in the data sheet. A 90.9 Ω resistor in series with 1000 Ω +3500 ppm/K does not give 1090 Ω +3333 ppm/K, but instead +3208 ppm/K. See also https://ez.analog.com/amplifiers/operational-amplifiers/f/q-a/572296/temperature-coefficient-in-ad538

Platin PT500 3850 ppm/K https://www.reichelt.de/platin-temp-sensor-22-x-4-0-mm-500-ohm-pcb-2240-pt500-p151245.html

Platin PT1000 3850 ppm/K https://www.reichelt.de/platin-temperatursensor-pt1000-klasse-a-f0-15--ertd2-pt-1000-a-p290780.html

KTY81-122 1000 Ω 0.79%/K = 7900 ppm/K https://www.reichelt.de/temperatursensor-ptc-1-kohm-55-150-c-kty-81-122-p9597.html

KTY 81-222 2000 Ω 0.79%/K = 7900 ppm/K https://www.reichelt.de/temperatursensor-ptc-2-kohm-55-150-c-kty-81-222-p9601.html

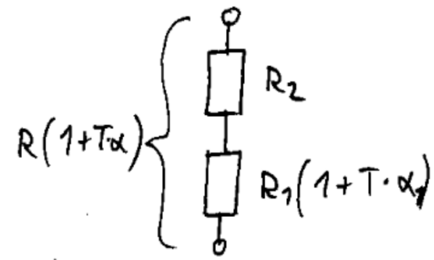
KTY 84-130 1000 Ω 0.62%/K = 6200 ppm/K https://www.reichelt.de/temperatursensor-40-300-c-kty-84-130-p9604.html

AN-304 Application Note: https://www.analog.com/media/en/technical-documentation/application-notes/51057130972814226449406767AN304.pdf

Considering Multipliers (Part 1): https://www.analog.com/en/analog-dialogue/articles/considering-multipliers-part-1.html

Analog Computation and Signal Processing: https://www.analog.com/media/en/training-seminars/design-handbooks/Linear-Design-Seminar-1987/Section5.pdf

These are the formulas for the resulting temperature coefficient alpha when two resistors are connected in series or parallel. R1 has temperature coefficient alpha1 and R2 doesn't have a temperature coefficient.



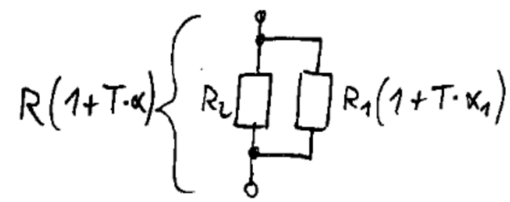
$$R = R_1 + R_2$$

$$\alpha = \alpha_1 \frac{R_1}{R_1 + R_2}$$

Calculate the unknown series resistor R2, if the resulting temperature coefficient alpha is known:
R2 = R1 * (alpha1 / alpha - 1)

Examples for AD538:
For PTC 1000 Ω +3500 ppm/K and alpha = +3333 ppm/K: R2 = 50.1 Ω
For PT1000 (alpha = +3850 ppm/K) and alpha = +3333 ppm/K: R2 = 155 Ω

Calculate the resulting temperature coefficient alpha if both resistors have temperature coefficients alpha1 and alpha2:
Series: alpha = (alpha1 * R1 + alpha2 * R2) / (R1 + R2)
Parallel: alpha = (alpha1 * R2 + alpha2 * R1) / (R1 + R2)

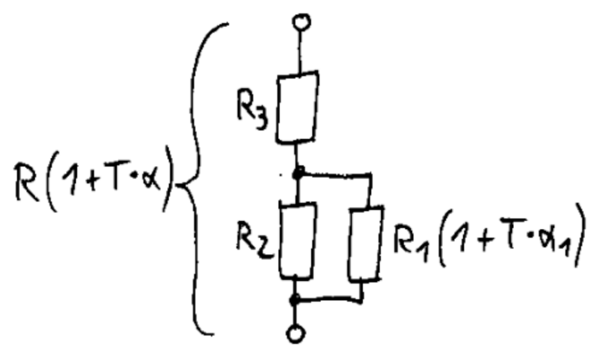


$$R = \frac{R_1 R_2}{R_1 + R_2}$$

$$\alpha = \alpha_1 \frac{R_2}{R_1 + R_2}$$

These are the formulas for the resulting temperature coefficient alpha when three resistors are connected. R1 has temperature coefficient alpha1 and R2 and R3 don't have temperature coefficients.

I don't know if it's analytically solvable for R2 and R3, if R, R1, alpha and alpha1 are given. If you find a solution, please let me know.



$$R = R_3 + \frac{R_1 R_2}{R_1 + R_2}$$

$$\alpha = \alpha_1 \frac{R_1 R_2^2}{(R_1 + R_2)(R_1 R_2 + R_1 R_3 + R_2 R_3)}$$

2.10 Diodes

The diodes in THAT are BAS170W Schottky diodes.

Typical forward voltage: 375mV at 1mA, 705mV at 10mA

Reverse current: < 0.1uA at 50V

Data sheet: https://www.mouser.de/datasheet/2/196/bas70_bas170series_2014-85076.pdf

https://www.infineon.com/dgdl/Infineon-BAS70series-DataSheet-v01_00-EN.pdf?fileId=5546d4626d66c2b1016d73f7ecf220d6

This reverse current may produce significant errors if the diode is connected to the summing junction of an integrator with a very long time constant.

1N4148:

Maximim forward voltage: < 1 V at 10mA (but typical forward voltage is smaller)

Reverse current: < 0.025uA at 20V

Data sheets: https://www.vishay.com/docs/81857/1n4148.pdf https://www.mouser.de/datasheet/2/308/1N914_D-2309448.pdf

BAS34:

70V, 200mA

Maximim forward voltage: < 1 V at 100mA (but typical forward voltage is smaller)

Reverse current: typical < 0.5nA at 30V Ultra small reverse current

Data sheet: https://www.vishay.com/docs/85541/bas33.pdf

The Z-Diodes in THAT are BZT52HC10. The Zener voltage is 10V +-0.5V.

The dynamic resistance is 70Ω at 1mA, or 15Ω at 5mA. The forward voltage is unspecified.

Data sheet: https://www.vishay.com/docs/86342/bzt52_series.pdf

ESD Protection diodes:

Vishay Semiconductors VCAN16A2-03S-E3-08

Usable voltage (without breakdown) up to 16.0V. The reverse breakdown voltage is in the 17.1V to 20.0V range.

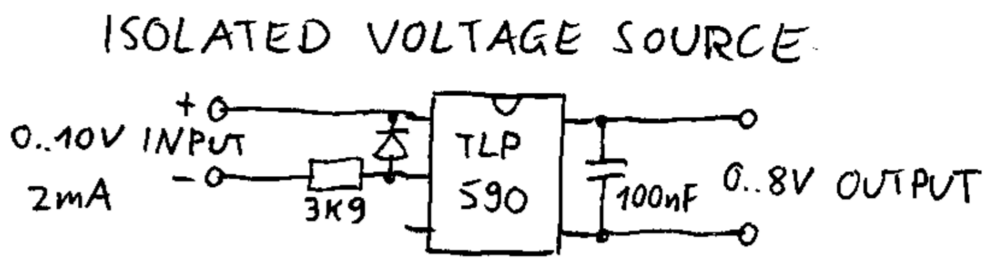
Data sheet: **https://www.vishay.com/docs/86175/vcan16a2-03s.pdf**

Idealized diode:

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 156ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

2.11 Isolated Voltage Source

Using the TLP590 photodiode array coupler, which has 16 photodiodes in series.



2.12 Analog Panel Meters

As an analog output device for THAT, it's best to use +-1mA meters with a 9.85kΩ series resistor (10k || 680k), so that 1 mA is equivalent to 1 machine unit (10V).

+-1mA meters: https://de.aliexpress.com/item/1005003385410991.html https://www.ebay.com/itm/184533810389

+-10V meter: https://de.aliexpress.com/item/1005005650339550.html

+-10V class 1.5 meter: https://de.aliexpress.com/item/1005005655298105.html

+-100µA meters: https://de.aliexpress.com/item/1005005650561266.html https://de.aliexpress.com/item/1005005625158185 https://de.aliexpress.com/item/1005004521461316

+-100µA class 1.5 meter: https://de.aliexpress.com/item/1005003467415360.html

SO-45 100mV meter: https://de.aliexpress.com/item/1005004347234374.html

Type 44C2 is DC meter, class 1.5, 100mm x 80mm

Type 85C1 is DC meter, class 2.5, 64mm x 56mm

Type 85L1 is AC meter, class 2.5, 64mm x 56mm

Type 91C4 is DC meter, class 2.5, 45mm x 45mm, most of these instruments don't have a screw for zero adjustment, you better use SO-45!

Type 91L4 is AC meter, class 2.5, 45mm x 45mm, most of these instruments don't have a screw for zero adjustment, you better use SO-45!

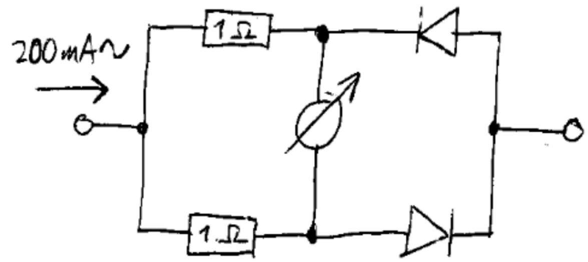
Type SO-45 is AC or DC meter, class 2.5, flange 47mm x 47mm, round 45mm. Available both as *Moving coil meter* which has a linear scale and as *Moving iron meter* which is for AC and DC and has a nonlinear scale.

Internal rectifier circuit in 200mA AC meter:

The advantage is that it has a linear scale.

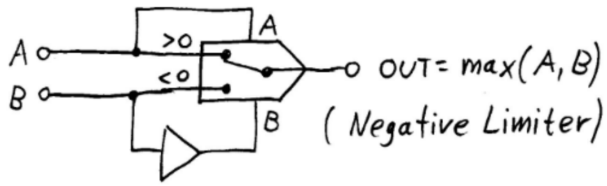
The disadvantage is 0.7V voltage drop in the measuring circuit.

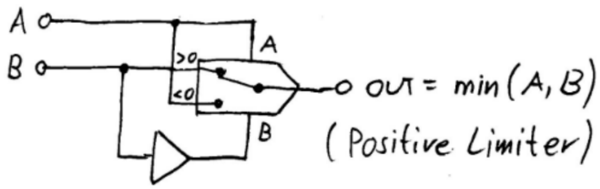
The diodes are 1N4007, but can be replaced by Schottky diodes for lower voltage drop.

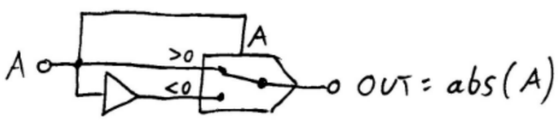


3 Functions

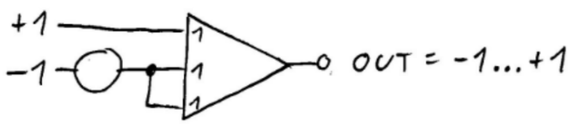
3.1 Helper Functions

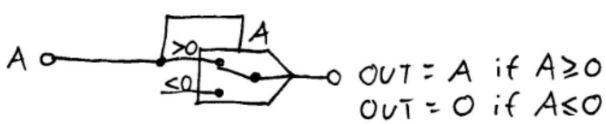


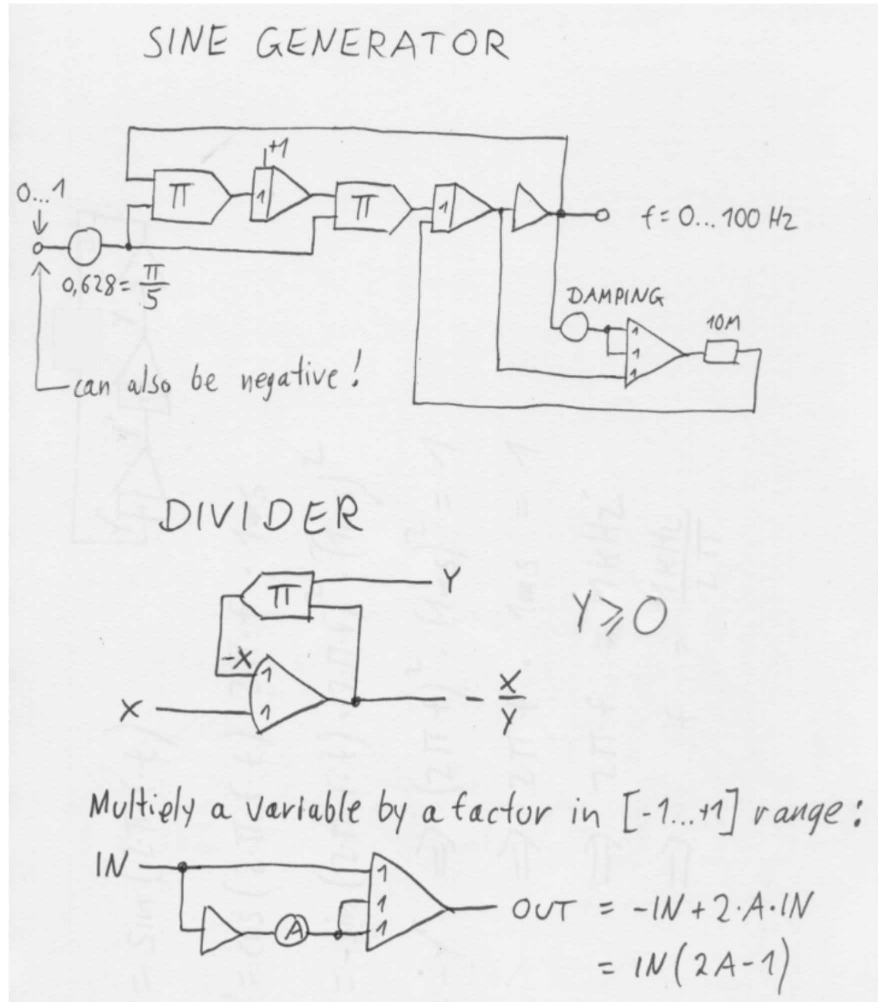




Make an output adjustable from -1 to +1:







3.2 Full Wave Rectifier

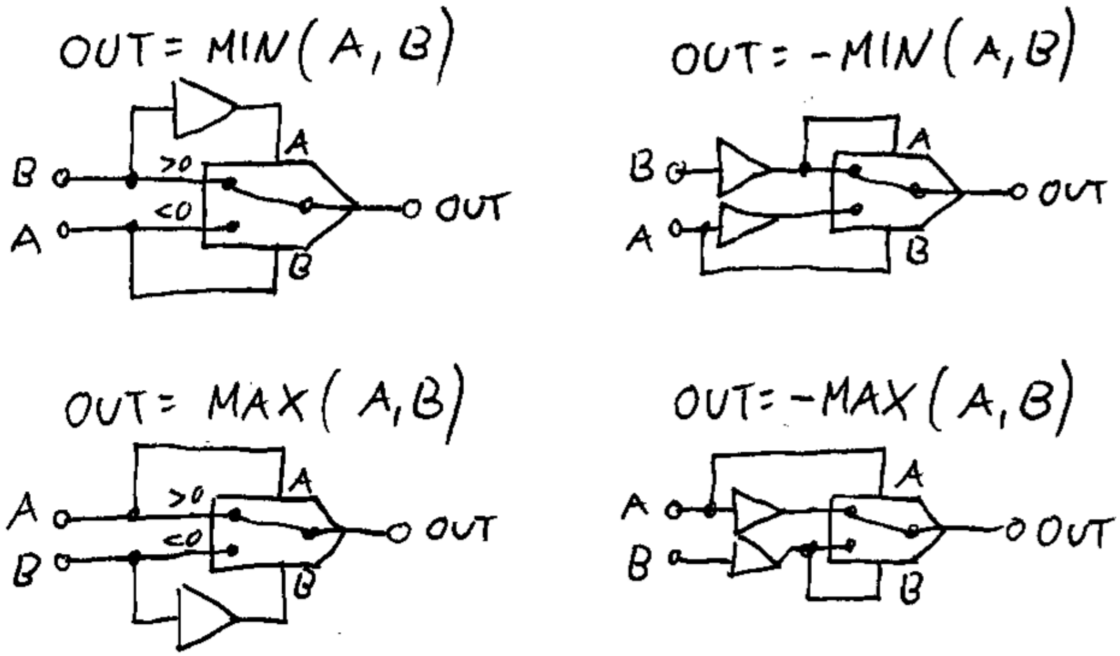
U. Tietze, Ch. Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 868

U. Tietze, Ch. Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 1204

Paul Horowitz, Winfield Hill: The Art of Electronics, Second Edition, page 221

Paul Horowitz, Winfield Hill: The Art of Electronics, Third Edition, page 257

3.3 Minimum and Maximum Functions

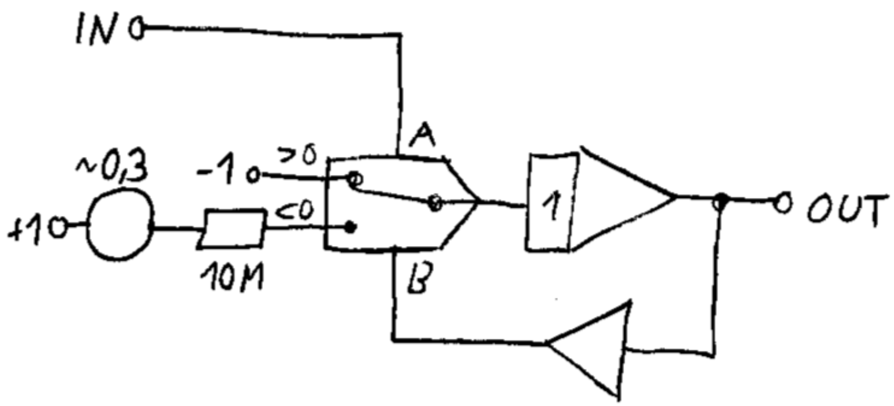


3.4 Measure the Peak Value of a Signal

This circuit compares the input signal with the output signal.

If output < input, the integrator gets -1 at its input, so that the output rises fast.

If output > input, the integrator gets a small positive signal at its input, so that the output falls slowly.



3.5 Limiting the Output of a Summer or Integrator

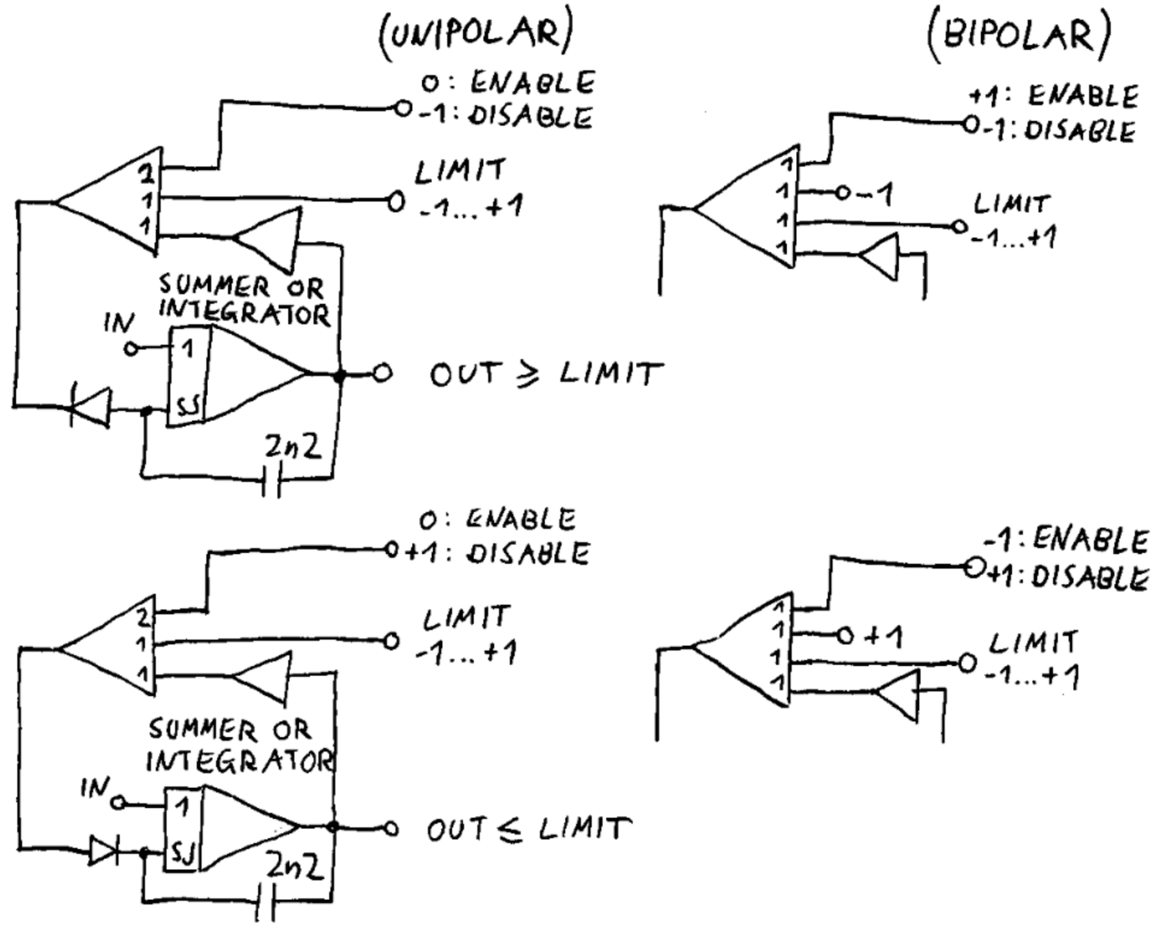
This works without comparators. When the limits are disabled, the output is not affected at all.

For summers the 2.2nF capacitor is required to remove oscillations. It must be so big because we have two amplifiers in the feedback loop.

For integrators the 2.2nF capacitor is not required.

The only difference between the two circuits is the polarity of the diode and the polarity of the disable signal.

The enable / disable input can be unipolar (left side) or bipolar (right side).

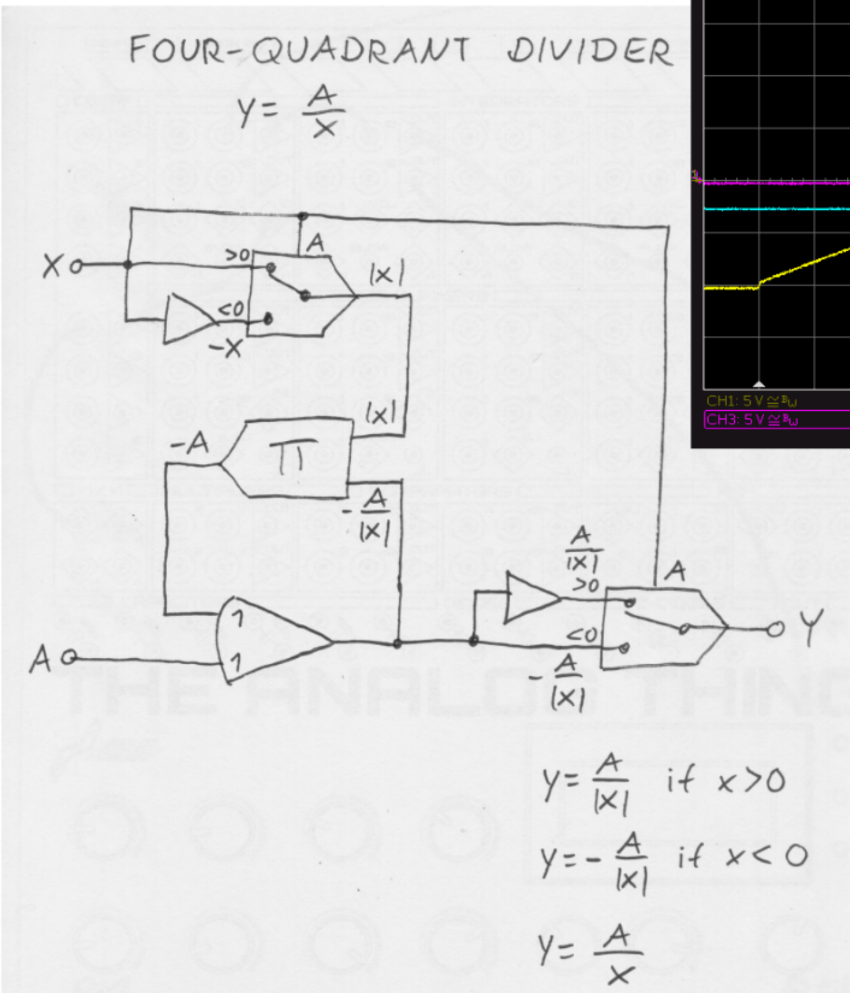


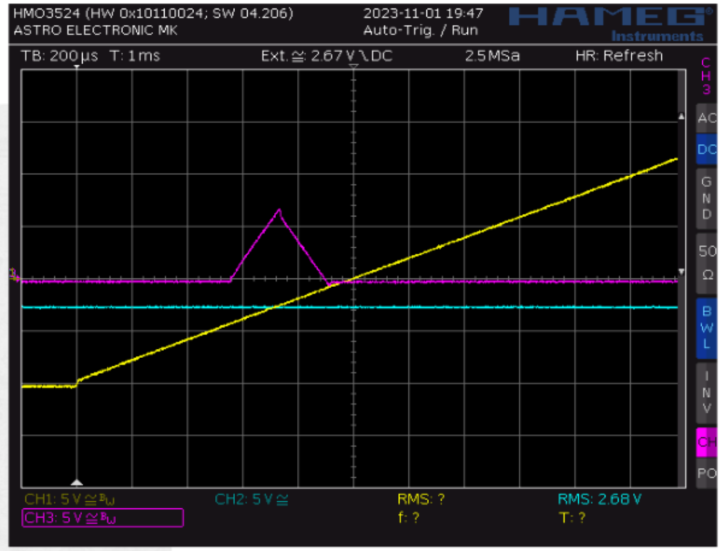
See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 143ff

Motion between end-stops:
Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 173ff

https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

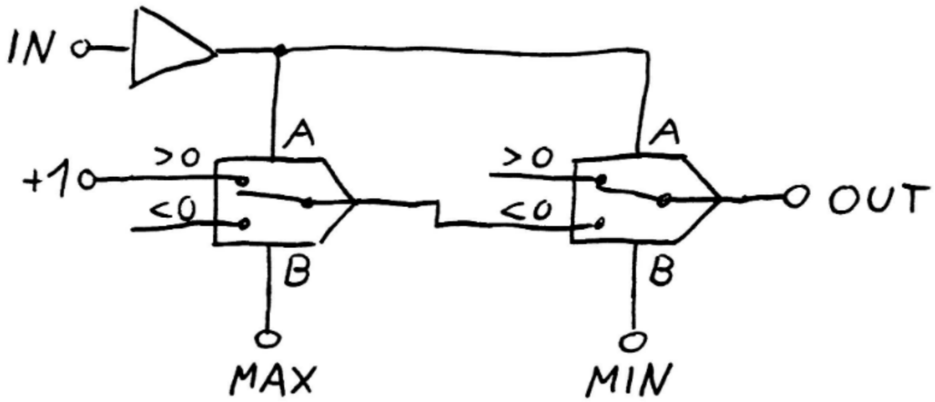
3.6 Four-Quadrant Division





3.7 Check if a Signal is between MIN and MAX

Check if IN is between MIN and MAX

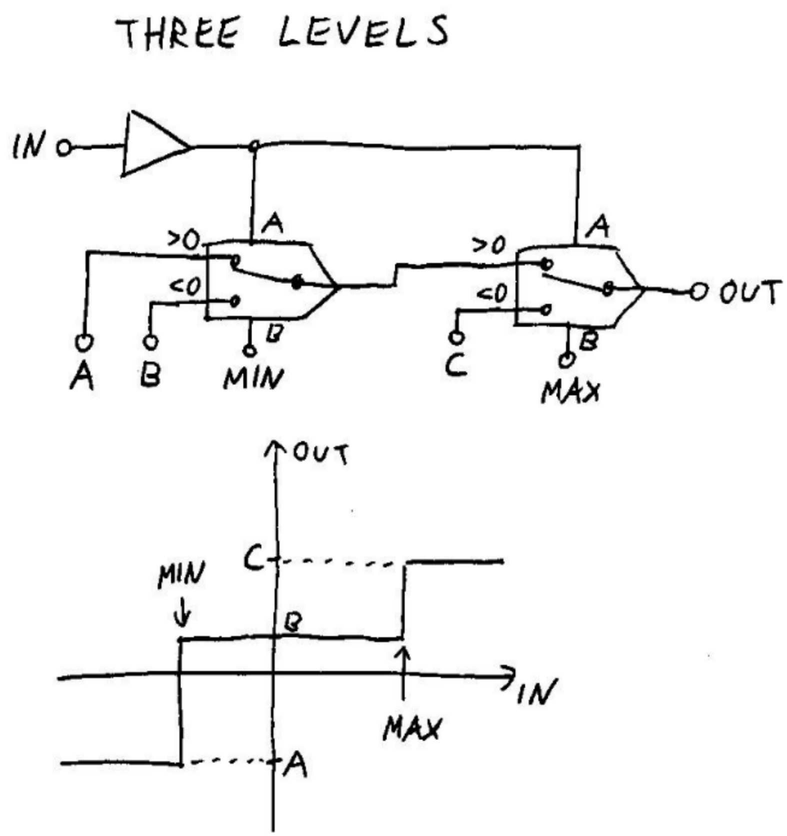


OUT = 1, if (IN > MIN) and (IN < MAX)

OUT = 0, if (IN > MAX) or (IN < MIN)

3.8 Three Levels

The output is A, B or C, depending if the input signal is below or above the MIN and MAX values. A, B and C can be functions of IN.



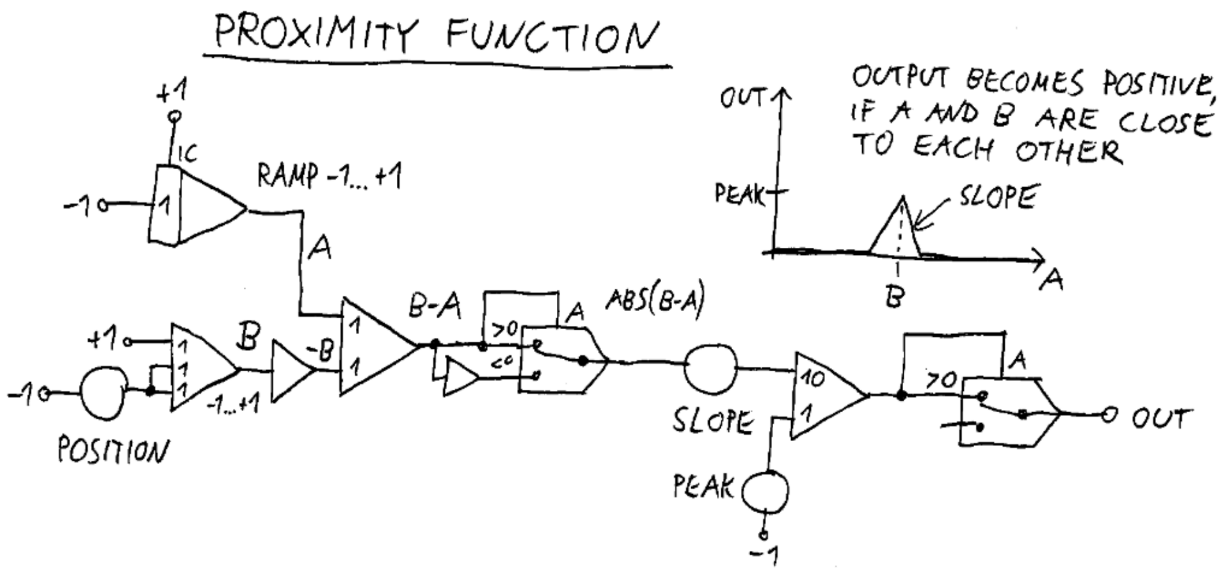
3.9 Proximity Functions

The output becomes positive, if the two inputs A and B are close to each other.
Peak level and slope are adjustable.

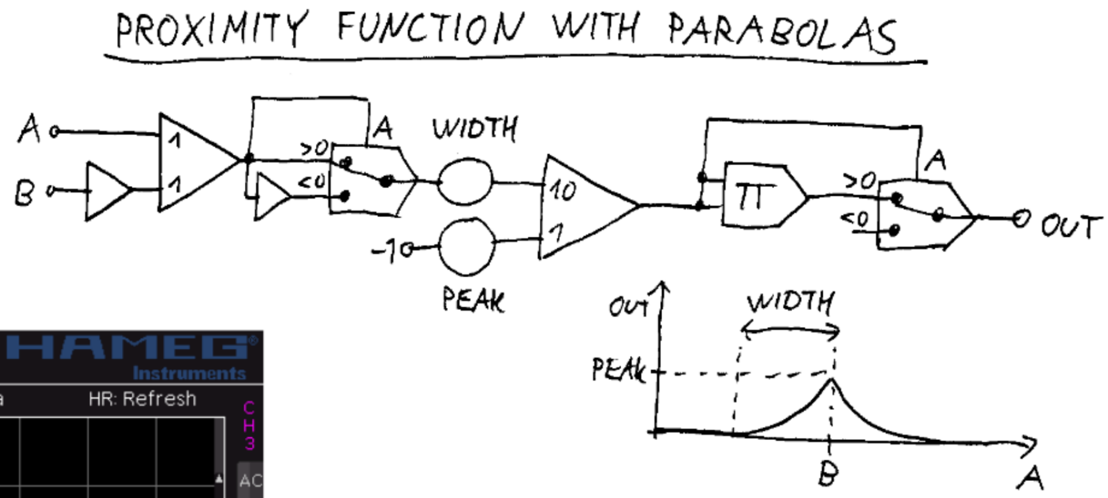
Yellow: Input A

Cyan: Input B

Magenta: Output



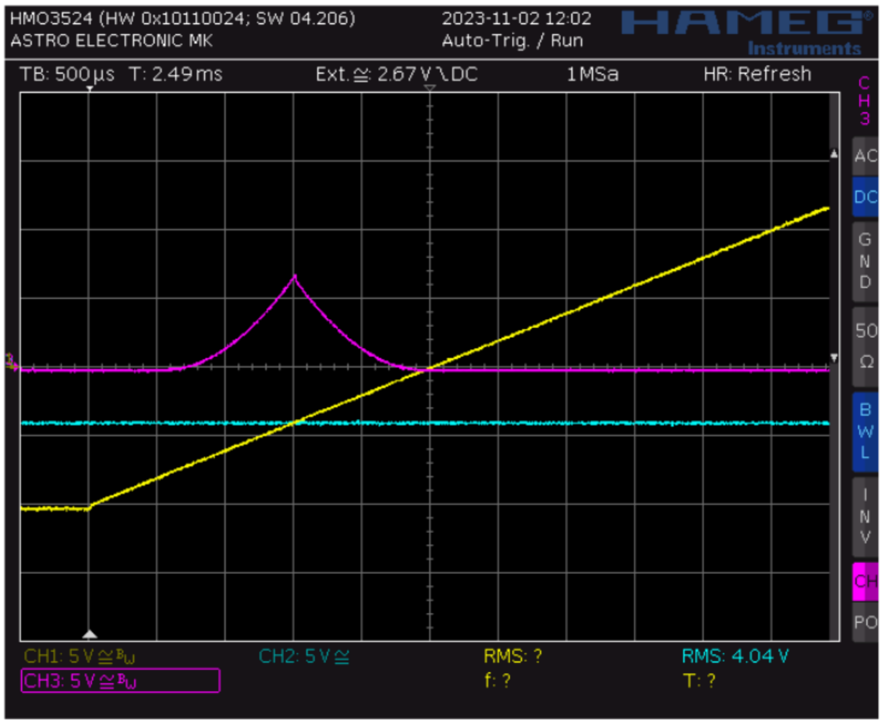
Proximity function with parabolas:



Yellow: Input A

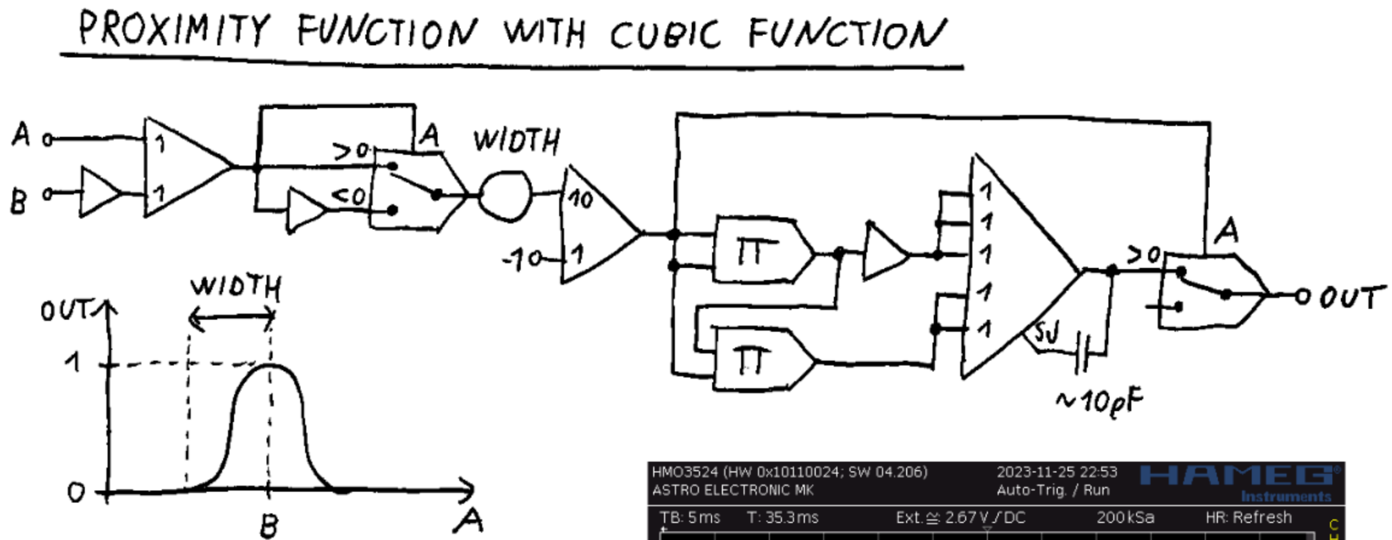
Cyan: Input B

Magenta: Output

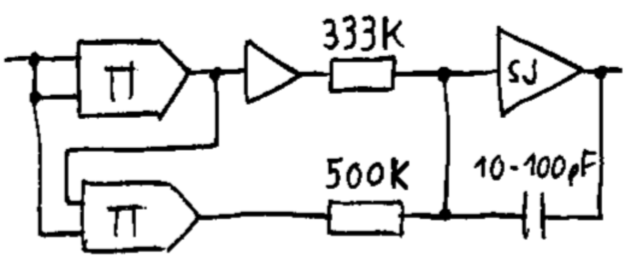


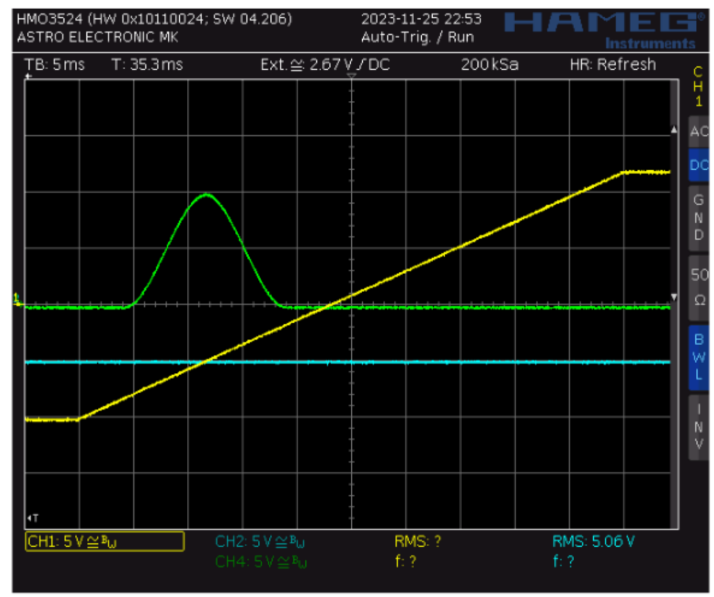
Proximity function with cubic function (approximated Gauss curve):

Note: For small width it's better to increase the feedback capacitor to 100pF.



The summer with 5 inputs can be replaced by an inverter with SJ input, if two additional resistors are used:

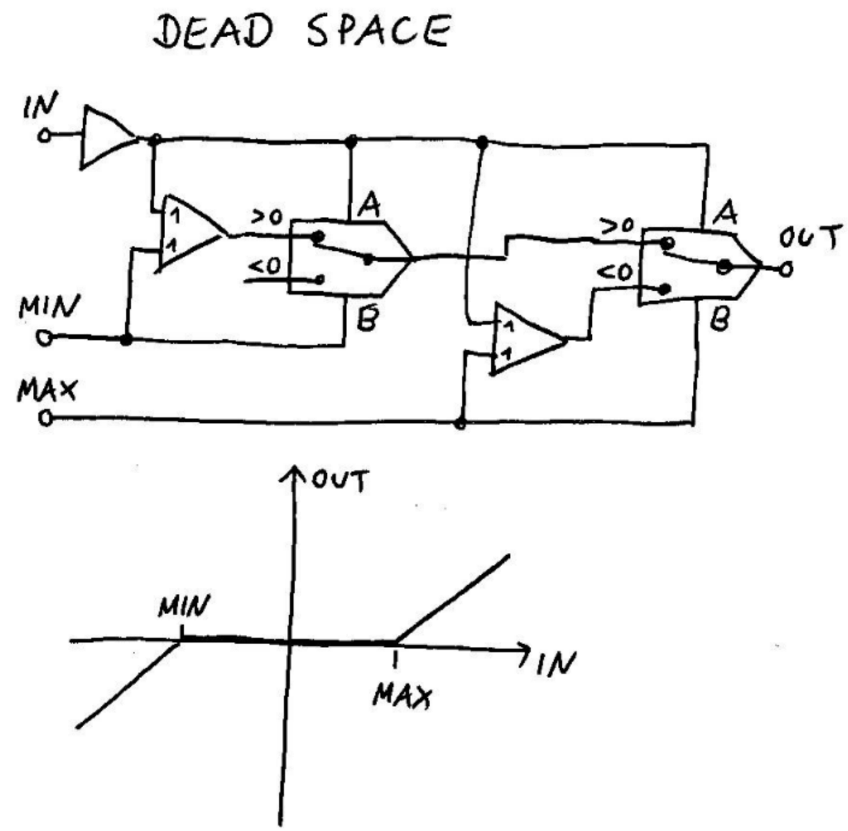




3.10 Dead Space

This is a dead space (dead band?) circuit. The same function can also be realized with two open amplifiers, one summer and 4 diodes, see figure 5.23 in Bernd's book.

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 161ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up



3.11 Backlash

This works, but who knows how to make the same thing with comparators?

See also: EAI Handbook, page 188 in the PDF:
https://analogmuseum.org/library/eai_handbook.pdf

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 171ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

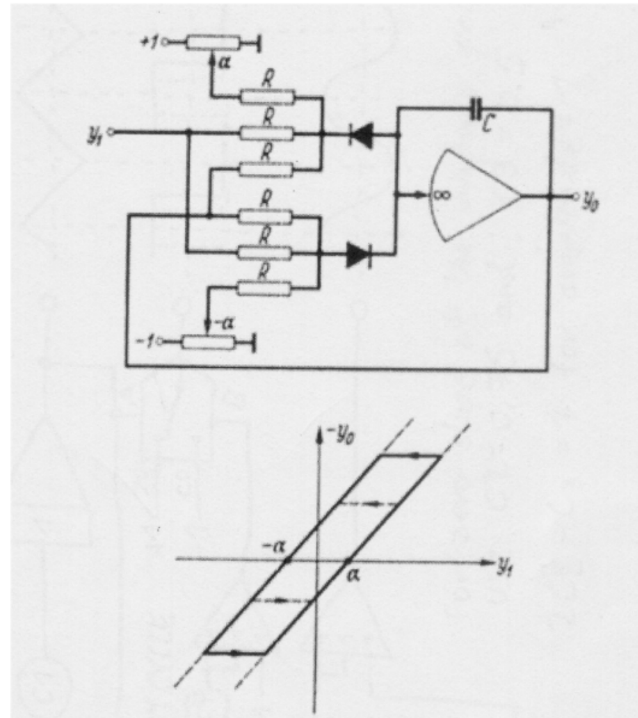


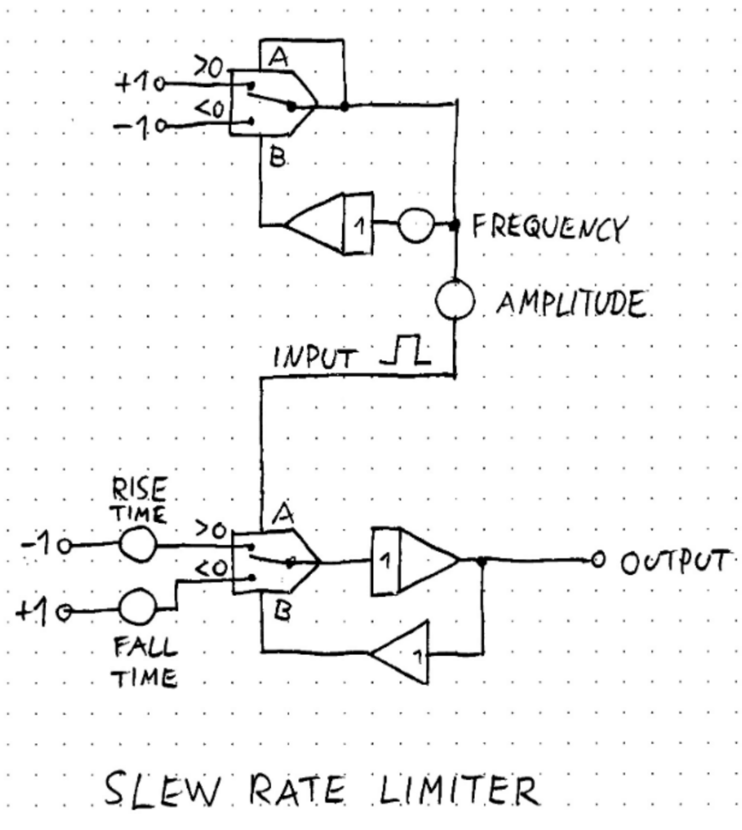
Abb. 3.2-35. Darstellung von Getriebelose (toter Gang).

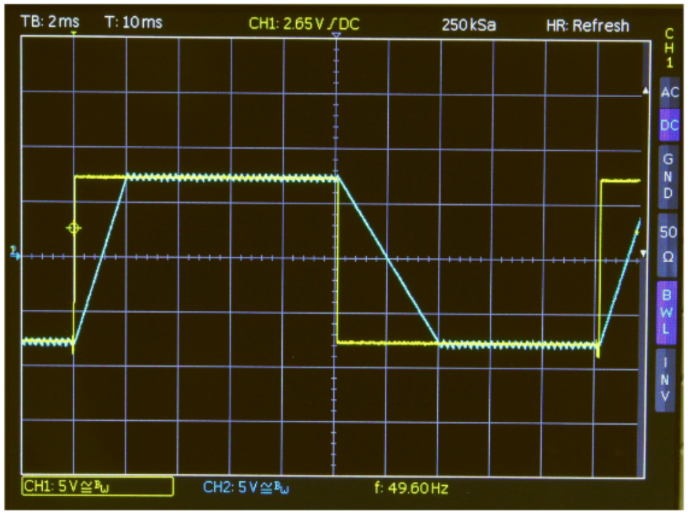
3.12 Coulomb and Static Friction

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 187ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

3.13 Slew Rate Limiter

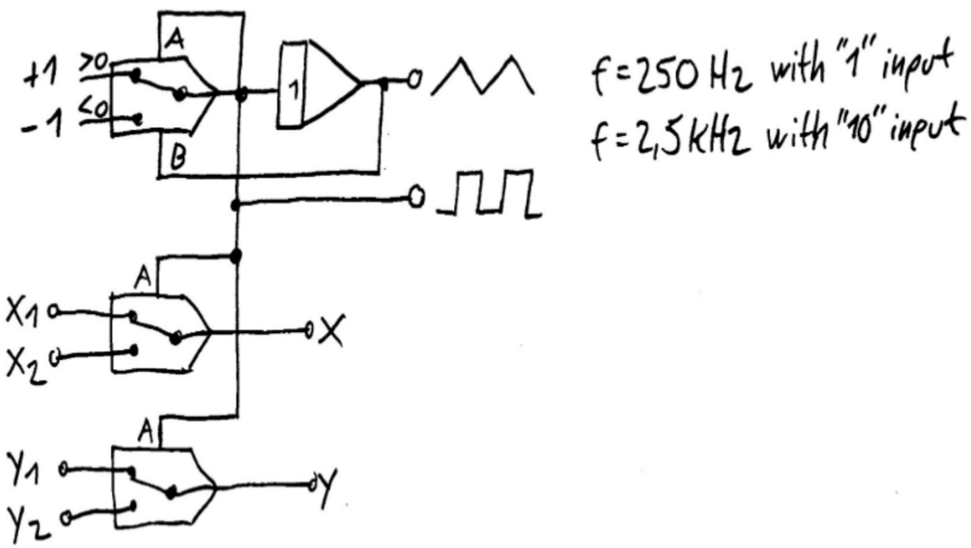
The upper half of this circuit is a simple square wave generator, which makes the yellow signal on the oscilloscope. The lower half is a slew rate limiter with adjustable rise and fall times. Its output is the blue signal on the oscilloscope.

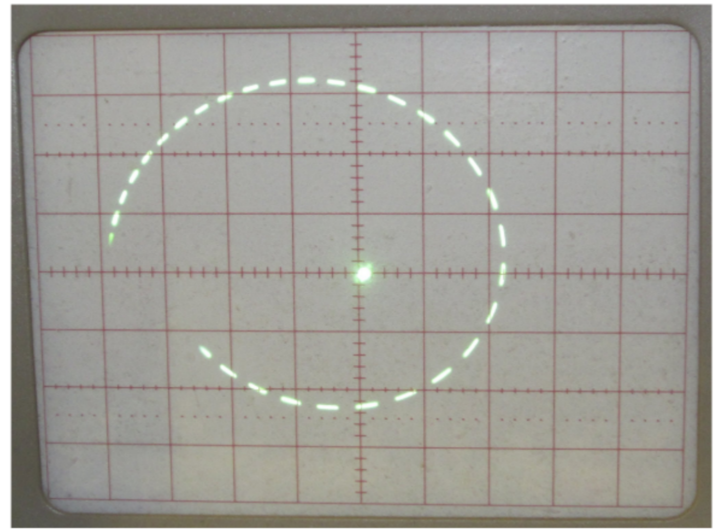




3.14 Multiplexer

This is a two-channel multiplexer for showing two dots on the x/y oscilloscope screen. The switching frequency can be selected by using the 1 or 10 input of the integrator. I did use the multiplexer for the planet orbit example, which I posted some time ago. One dot for the planet and the other for the sun in the center. With the low switching frequency (250Hz), you can see a dotted line, with the planet's speed proportional to the dot length. With the fast switching frequency it becomes a solid line.





3.15 Low Pass Filter

Integration Resistor	Integration Capacitor	Integration time constant k_0
100 kΩ ("10" Input)	1nF (normal)	100 µs
1 MΩ ("1" Input)	1nF (normal)	1 ms
1 MΩ ("1" Input)	10nF (external)	10 ms
1 MΩ ("1" Input)	100nF (SLOW)	100 ms
10 MΩ (external)	100nF (SLOW)	1 s
100 MΩ (external)	100nF (SLOW)	10 s
1 GΩ (external)	100nF (SLOW)	100 s
1 GΩ (external)	1µF (with open amplifier)	1000 s
1 GΩ (external)	10µF (with open amplifier)	10000 s

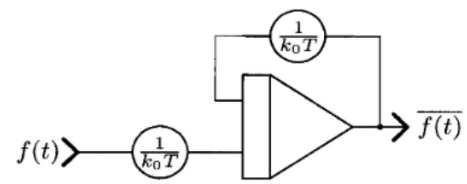


Fig. 5.11. Basic circuit of a low pass filter

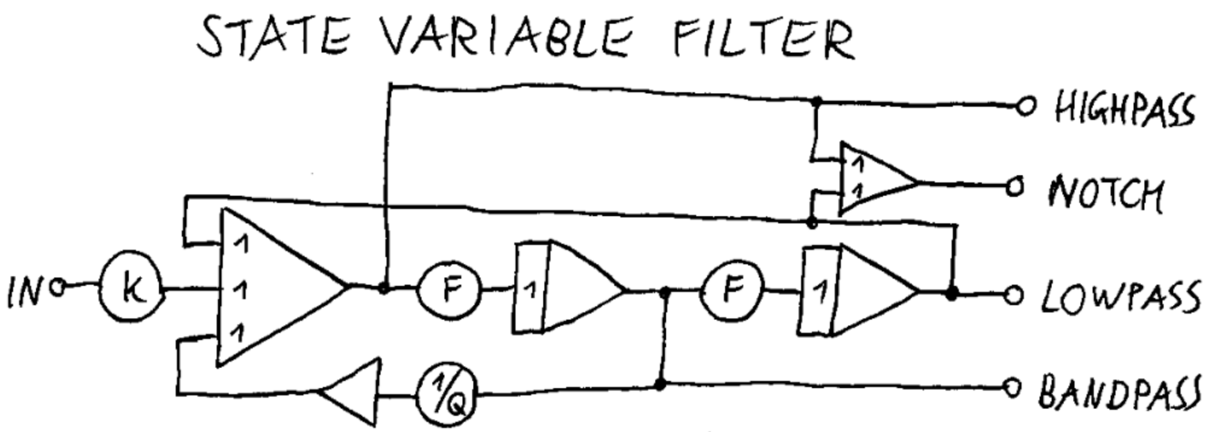
Source of the image: Bernd Ulmann: Analog and Hybrid Computer Programming, page 83

$f_{FILTER} = K / (2\pi * k_0)$ where K are the two coefficients (at the input and in the feedback loop) and k_0 is the time constant of the integrator

U_{OUT} / U_{IN} = 1 / sqrt(1 + (f_{IN} / f_{FILTER})^2)

f_{IN} / f_{FILTER}	U_{OUT} / U_{IN}
0.1	0.995
0.25	0.970
0.5	0.894
1.0	0.707
2.0	0.447
4.0	0.243
10.0	0.100

3.16 State Variabe Filter (Lowpass, Highpass, Bandpass, Notch)



Center frequency $f_0 = F / (2\pi * k_0)$, where k_0 is the time constant of the integrators.

The bandwidth is $BW = f_2 - f_1$. The center frequency is $f_0 = sqrt(f_1 * f_2)$. The quality factor is $Q = f_0 / BW$.

For lowpass, highpass and notch filter the gain is equal K.

For bandpass filter the gain at the center frequency is equal K*Q. That means K should be smaller or equal 1/Q, so that the output can't overflow when the input uses the full machine unit range.

A notch filter can be realized by summing the lowpass and highpass outputs of a state variable filter. To compensate for the small gain (if K is small), you can use the "10" inputs of the summer for the notch signal.

If you want to control the filter's center frequency by a voltage, you can simply replace both "F" coefficients by multipliers. But keep in mind that the bandwidth is proportional to the center frequency. So it's not a good idea to build a spectrum analyzer with this filter (unless the frequency span is very small).

See also chapter 11.7 in this book:
https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Operational_Amplifiers_and_Linear_Integrated_Circuits_-_Theory_and_Application_%28Fiore%29/11%3A_Active_Filters/11.07%3A_Band-Pass_Filter_Realizations

Formulas for the bandpass filter:

Gain: $U_{OUT} / U_{IN} = (K * \Omega) / \mathrm{sqrt}(1 + \Omega^2 * (1 / Q^2 - 2) + \Omega^4)$

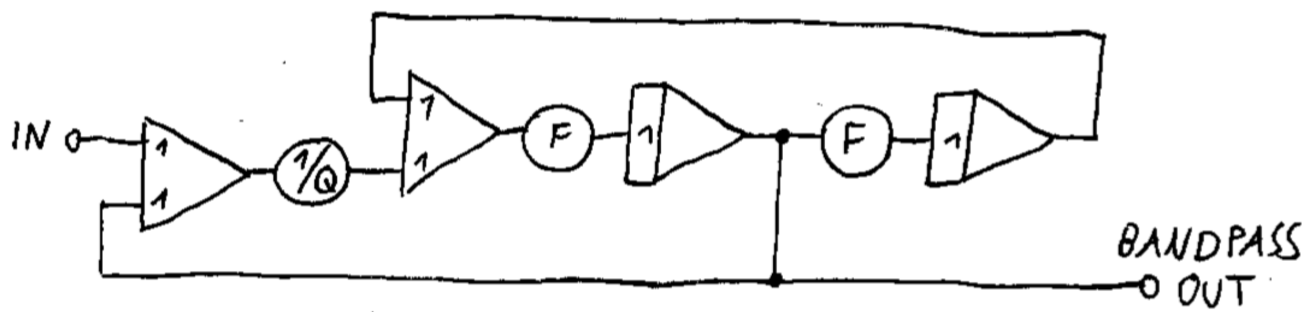
Phase shift: $\varphi = \mathrm{atan}(Q * (1 - \Omega^2) / \Omega)$

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 426

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 872

The drawback of the previous circuit is that the 1/Q coeffiecient does not only change the quality factor, but also the gain at the center frequency. This can be compensated by simultaneously changing also the K coefficient.

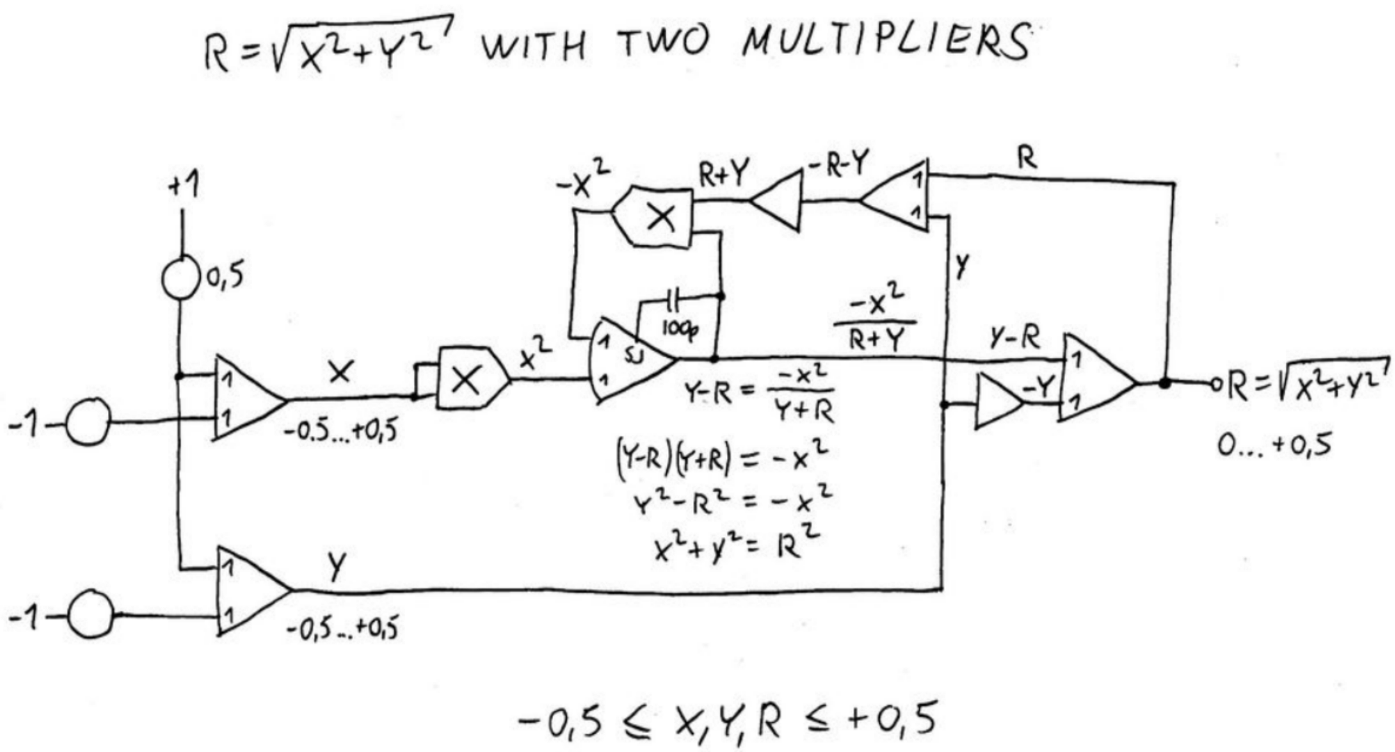
In the following circuit the quality factor can be adjusted with one coefficient. The gain at center frequency is always 1.



3.17 Vector Length with two Multipliers

Calculate the length of a two-dimensional vector with two multipliers. It's important that the X and Y signals must not exceed the -0.5 to +0.5 range. Otherwise the -R-Y and R+Y signals would overflow.

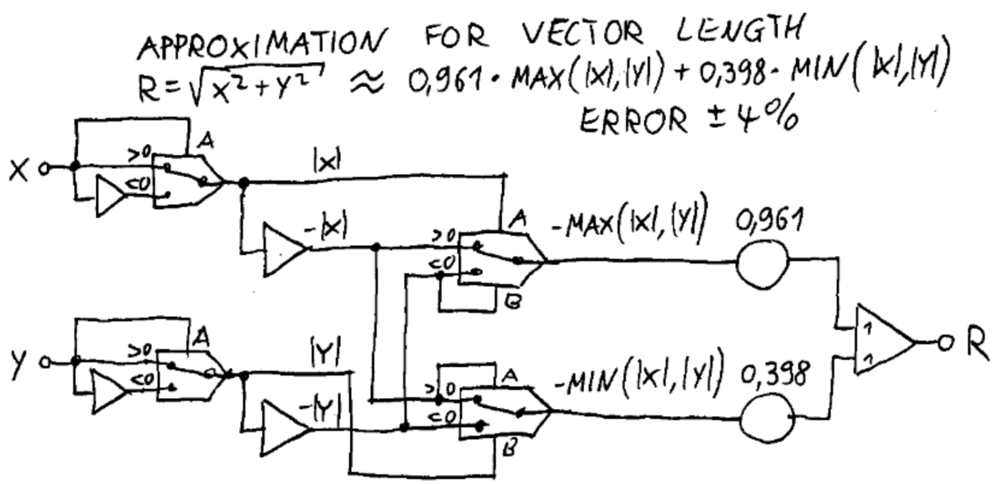
I found the example here: http://www.analogmuseum.org/library/pythagoras.pdf



See also: https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf#page=3

3.18 Approximations for Vector Length

Approximation for $r = sqrt(x^2 + y^2)$	Error range for a point on the unit circle r = 1	Error in %	Required components in THAT
r ≈ max(|x|,|y|)	0.707...1.000 largest error at 45°	-29.3%...+0%	3 Comparators 3 Inverters
r ≈ max(|x|,|y|) + 0.414 * min(|x|,|y|) Note: 0.414 = sqrt(2) - 1	1.000...1.082 largest error at 22.5°	-0.0%...+8.2%	4 Comparators 4 Inverters 1 Coefficient 1 Summer
r ≈ max(|x|,|y|) + 13/32 * min(|x|,|y|)	0.994...1.079 largest error at 22.1°	-0.6%...+7.9%	same as above, but optimized for 8-bit 8051 microcontroller
r ≈ 0.960 * max(|x|,|y|) + 0.398 * min(|x|,|y|)	0.960...1.039 balanced positive and negative errors at 0°, 22.5° and 45°	-4.0%...+3.9%	4 Comparators 4 Inverters 2 Coefficients 1 Summer



See also: Bronstein-Semendjajew: Taschenbuch der Mathematik, 21. Auflage, Page 101

3.19 Function Generators

See chapter 2.6 in Bernd Ulmann: Analog and Hybrid Computer Programming

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 9. Auflage, page 345ff

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 797ff

User-defined transfer functions can easily be realized with hybrid analog / digital circuits, see my example with Teensy LC.

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 165ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

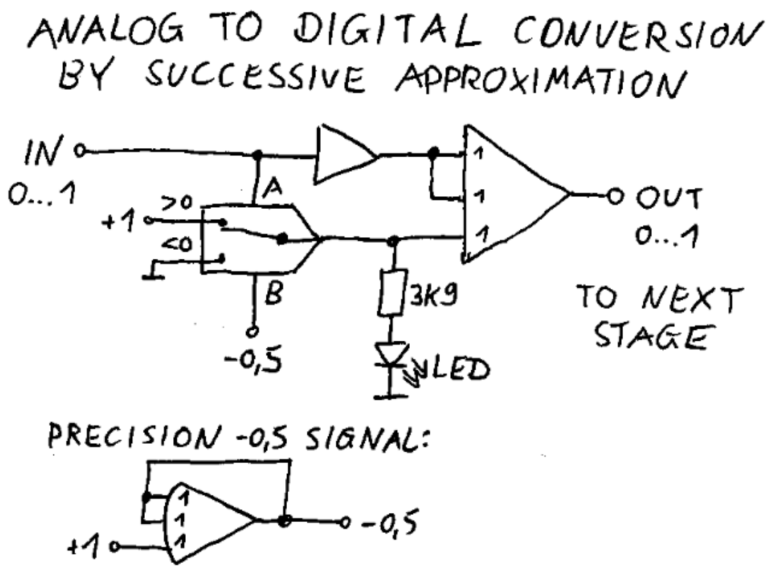
3.20 Analog to Digital Conversion by Successive Approximation

This circuit can be cascaded. Because each bit requires a comparator, only two bits are possible with one THAT. With two THATs you can realize a 4-bit converter, and so on.

Each stage works as follows:

- **If the input is less than 0.5, the digital bit is 0 and the output is the input multiplied by two.**
- **If the input is greater or equal 0.5, the digital bit is 1 and the output is the input multiplied by two, minus one.**

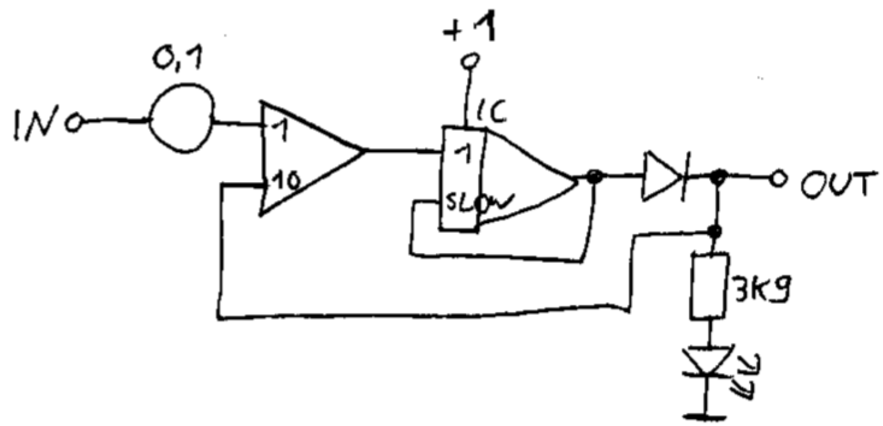
The output of the last stage can be divided by a suitable power of two and then becomes the remainder.



The advantage of the precision -0.5 signal is that it doesn't need an adjustment with a coefficient.

3.21 Switch on Lights, one after the other

Connect the output to the input of the next stage. The first stage gets +1 as input.

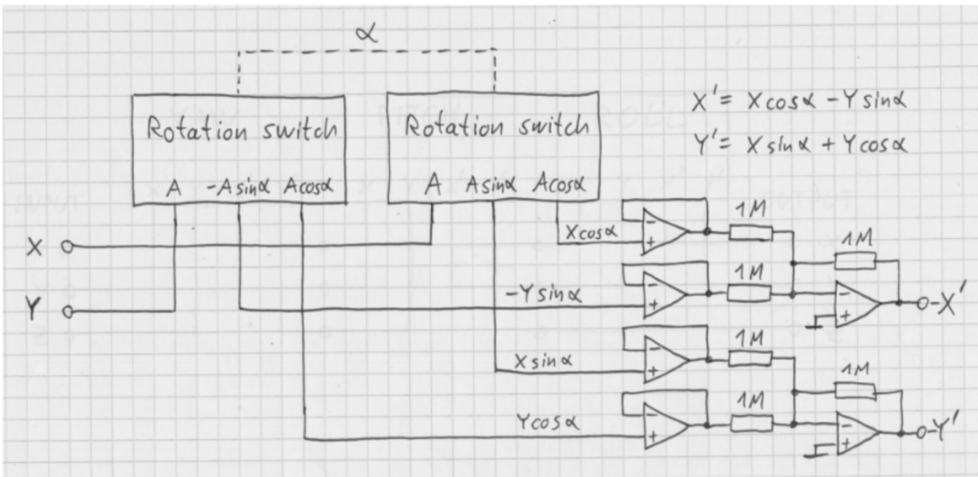


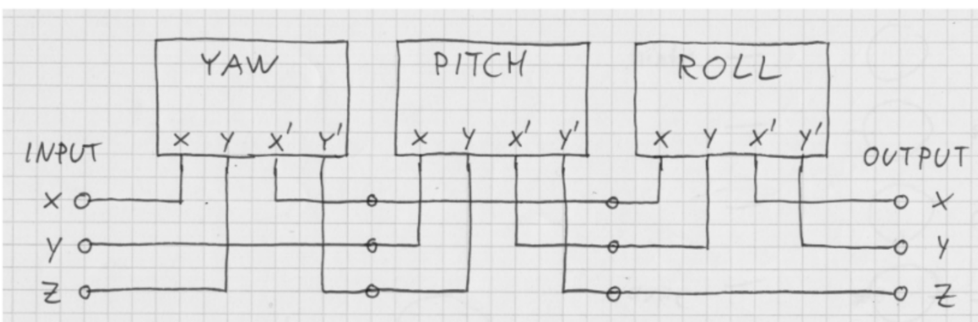
3.22 Voltage to Time Conversion

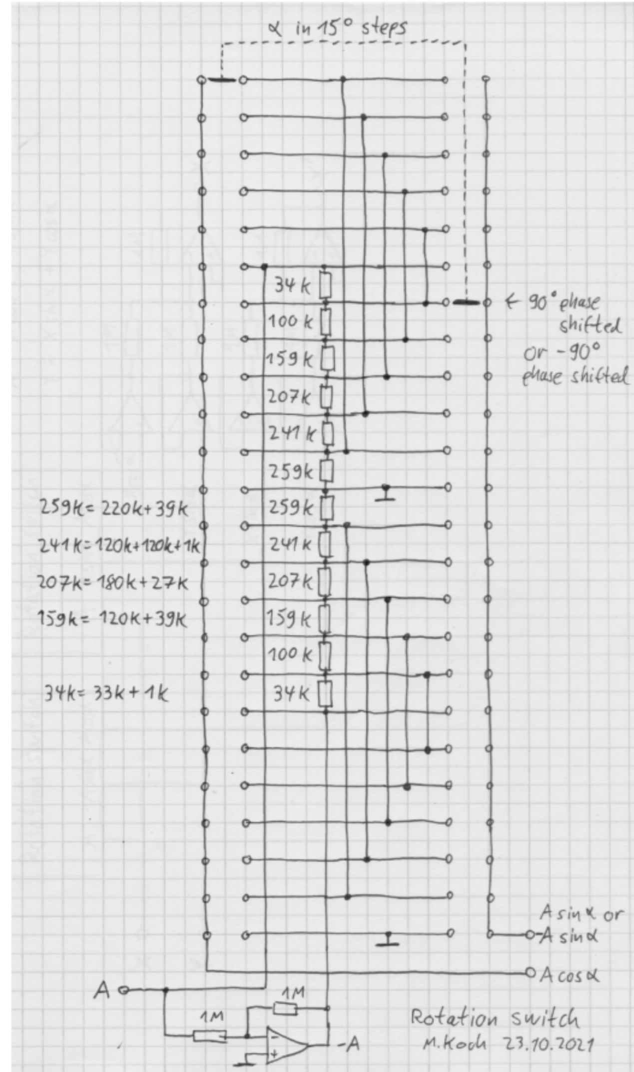
- **If a variable control voltage is connected to the input of an integrator and we stop the integration when a fixed output voltage U_{max} is reached, the required time is inversely proportional to the input voltage. With other words: Voltage is proportional to frequency.**
- **If a constant is connected to the input of an integrator and we stop the integration when the variable control voltage is reached, the required time is proportional to the input voltage.**

3.23 Rotation Matrix in 3D Space

4x24 rotary switches are available from Aliexpress:
https://de.aliexpress.com/item/1005001651673638.html







4 Hybrid Analog / Digital Computing

4.1 Pinout of THATs Hybrid Port

Wiki: https://the-analog-thing.org/wiki/Hybrid_Computer

Schematic: https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic5.pdf

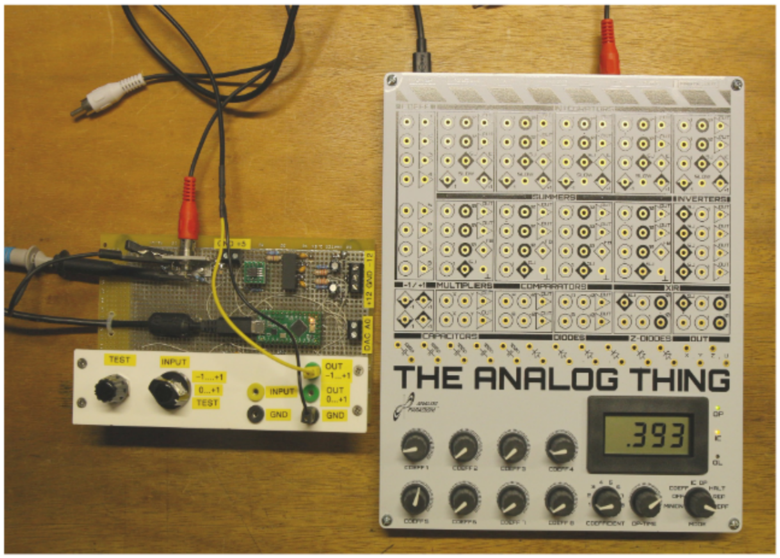
Hybrid port description: https://the-analog-thing.org/w/images/7/78/THAT_hybrid_port_desc.pdf

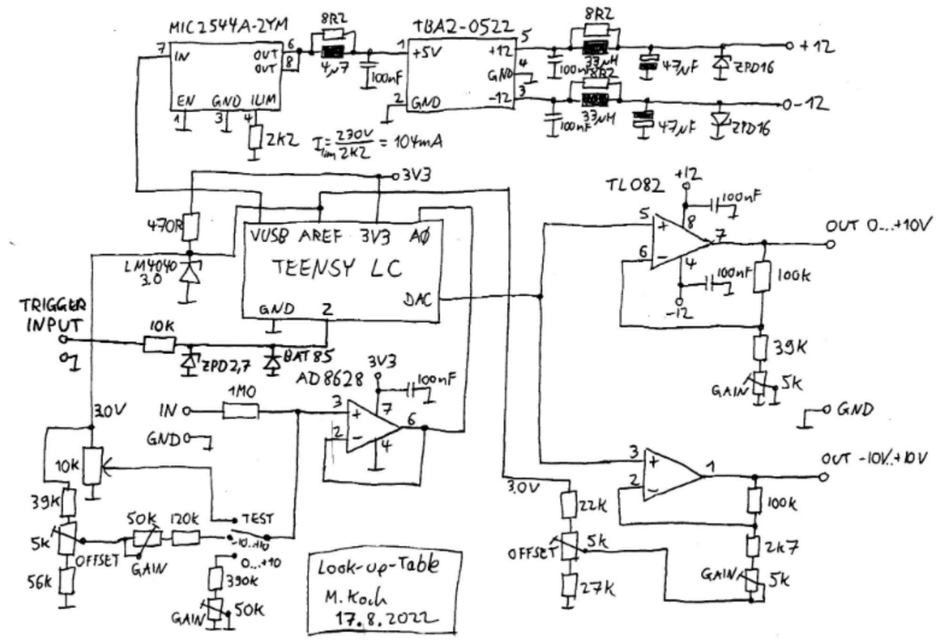
Pin	Name	Function
1 3 5 7	IN-X IN-Y IN-Z IN-U	Same signals as cinch connector X, voltage is 0.1 * OUT-X (OUT-Y, OUT-Z, OUT-U), range -1V to +1V, output impedance is 423 Ohm Might also be used as input, but sinks 21.1 mA at 1 MU, input impedance is 470 Ohm
2 4 6 8	HYB-X HYB-Y HYB-Z HYB-U	Low impedance analog output signals, voltage is 0.1 * OUT-X + 1.64V, range 0.64V to 2.64V for +-1 MU
9, 10	GND	Ground
11, 12	VUSB	Is connected to +5V USB supply voltage via 750mA PTC fuse
13	DIR	This is an input with 3.3V level, low level enables THATs hybrid mode. The input has an internal pullup resistor to 3.3V and it not 5V tolerant.
14	-Mode_OP	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = operate In hybrid mode: 5V tolerant 3.3V input, sets THATs operation mode, the input has no internal pullup resistor
15	Voffset	+1.64V
16	-Mode_IC	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = initial condition In hybrid mode: 5V tolerant 3.3V input, sets THATs initial condition mode, the input has no internal pullup resistor

Note: Looking from the rear side into the hybrid port: Pin 1 is top right, Pin 2 is bottom right.

4.2 Teensy LC

My "Teensy LC" board for hybrid analog / digital computing is in detail described in this book: http://www.astro-electronic.de/Teensy_Book.pdf





See also: https://the-analog-thing.org/wiki/Hybrid_Computer

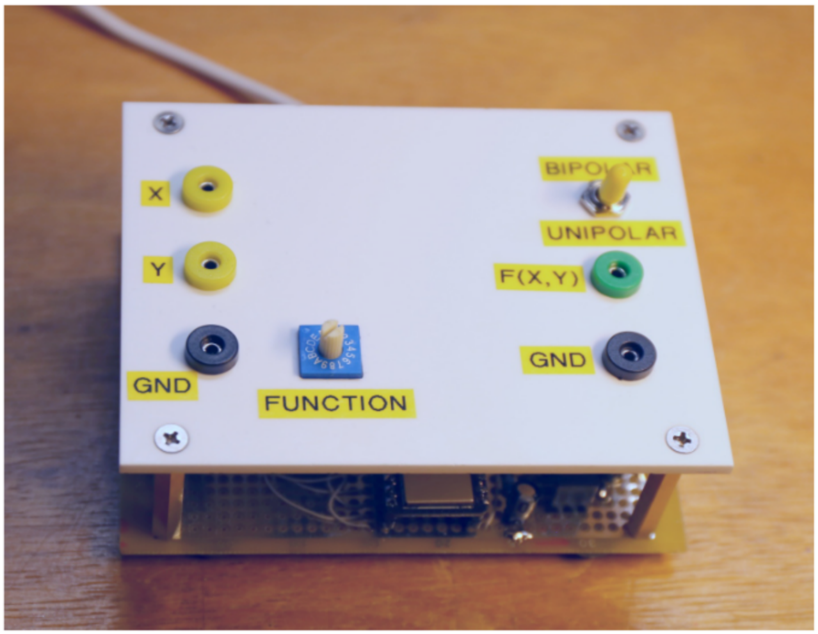
See also: https://www.mdpi.com/1424-8220/23/7/3599

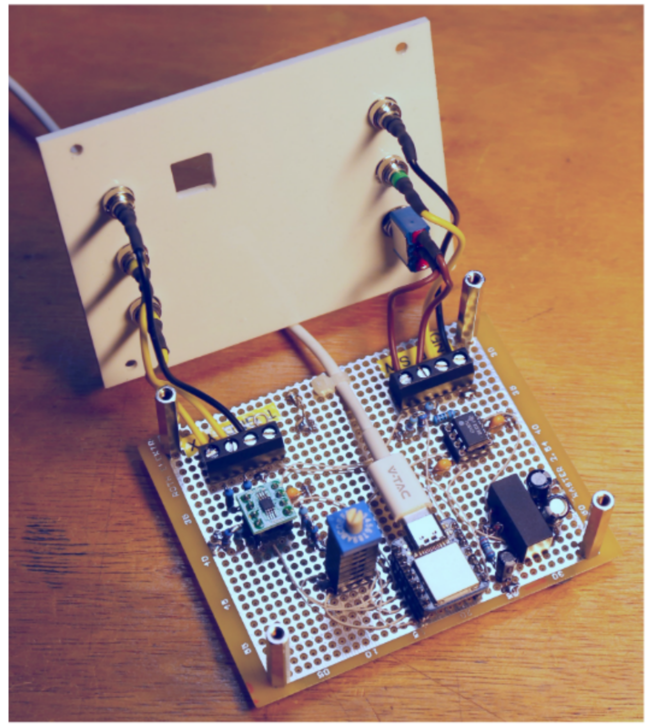
4.3 Function Generator with Seeed XIAO SAMD21

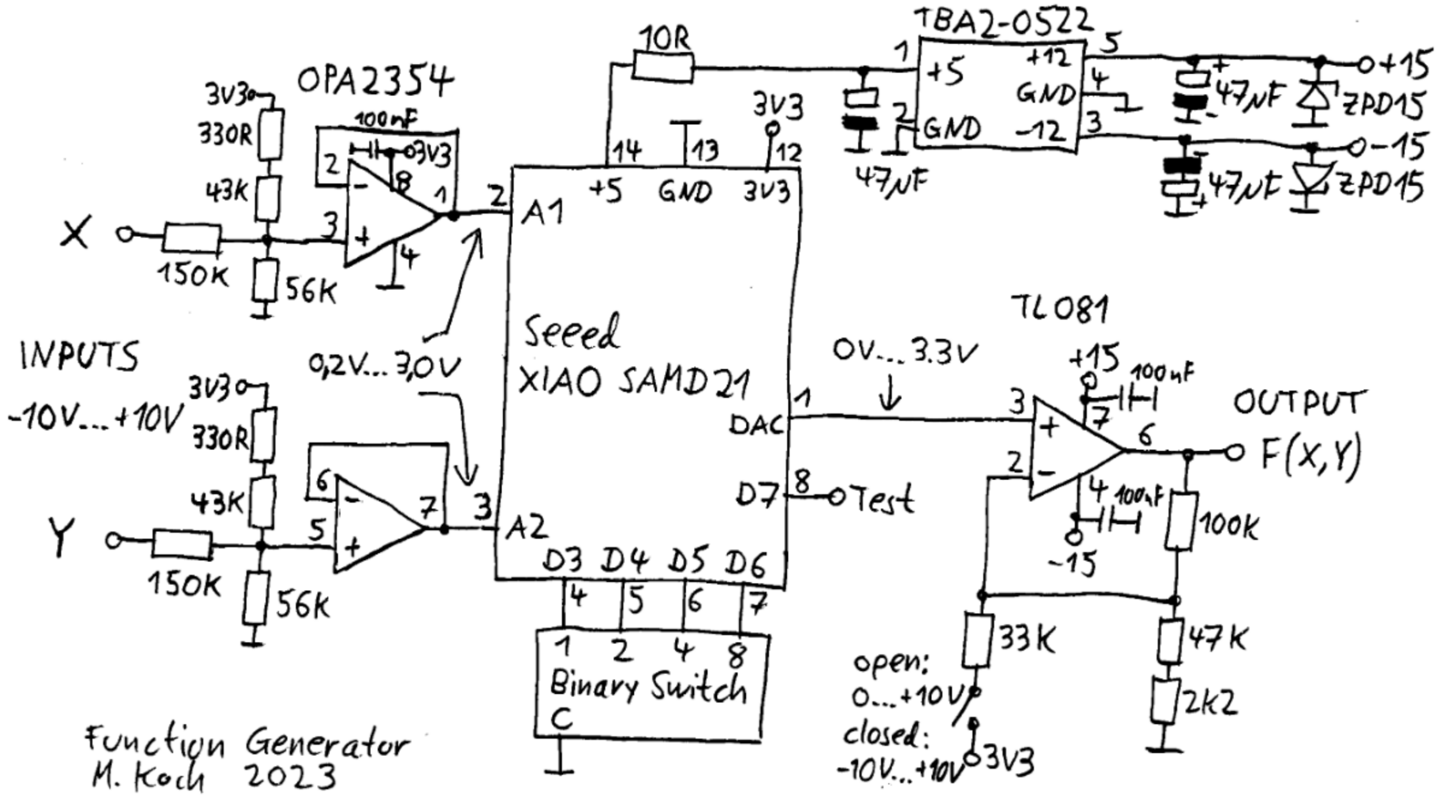
This is a small function generator with the Seeed XIAO SAMD21 microcontroller. It has two analog -10V to +10V inputs (+-1 machine unit) and one analog output which can be switched to unipolar mode (0 to +10V) or bipolar mode (-10V to +10V). 16 different functions F(X,Y) can be selected with a binary switch. The software doesn't use lookup tables. The functions are calculated in realtime. The samplerate is between 3 and 10 kHz. I did try to make the circuit as simple as possible. There are no potentiometers for offset and input gain. That can be adjusted in software.

List of functions:
x*y, x^2, x^3, sqrt(x), x^(1/3), sgn(x), x/(10y), sqrt(x*x+y*y), x^(5y), exp(x-1), exp(2x-2), 1+ln(x), 1+0.5*ln(x), atan2(y/x)/pi, sin(x*pi), cos(x*pi)

But of course you can change the functions in software.







Although the OPA2354 has rail-to-rail inputs and outputs, it wasn't able to drive the analog input of the microcontroller close enough to the 3.3V rail. That's why I'm using the voltage range from 0.2V to 3.0V. The ADC has 12-bit resolution and the input range is from 0 to 3.3V. That means I'm using here not the full 12 bits. The effective resolution is $\log_2(4096 * 2.8V / 3.3V) = 11.7$ bits.
The DAC has only 10-bit resolution and the full 0 to 3.3V range is used. The resolution is about 20mV in bipolar mode or 10mV in unipolar mode.

Note: The 10R resistor at the input of the voltage converter was inserted to make sure that the current can never become larger than 0.5 A. This resistor is unnecessary because the voltage converter has an internal overload protection.
With shorted +-12V outputs, the input current is less than 0.2 A. With worst case load (about 56 Ohms at each output), the input current is less than 0.8 A.

```
/*
  Programmable Function Generator for THAT with Seeed_XIAO
  Michael Koch, December 2023
*/

bool bipolar;
int mode;
float x,y,f;


// the setup function runs once when you press reset or power the board
void setup() {
  //pinMode(LED_BUILTIN, OUTPUT);
  pinMode(A1,INPUT);           // analog inputs
  pinMode(A2,INPUT);
  pinMode(D3, INPUT_PULLUP);   // pullup resistors for mode switch inputs
  pinMode(D4, INPUT_PULLUP);
  pinMode(D5, INPUT_PULLUP);
  pinMode(D6, INPUT_PULLUP);
  pinMode(D7, OUTPUT);         // Test pin for sample rate
  analogReadResolution(12);
  analogWriteResolution(10);
}

// the loop function runs over and over again forever
void loop() {
  mode = 15 - digitalRead(D3);  // read the mode switch
  mode -= 2 * digitalRead(D4);
  mode -= 4 * digitalRead(D5);
  mode -= 8 * digitalRead(D6);

  // You can fine-tune the offset and gain values!
  digitalWrite(D7, HIGH);                // test pin = high
  x = -1.15 + 0.00058 * analogRead(A1);  // read analog x input
  digitalWrite(D7, LOW);                 // test pin = low
  y = -1.15 + 0.00058 * analogRead(A2);  // read analog y input

  switch (mode)
  {
    case 0:  f = x * y;
             bipolar = true;
             break;
```

```
case 1:  f = x * x;
         bipolar = false;
         break;
case 2:  f = x * x * x;
         bipolar = true;
         break;
case 3:  f = (x>0) ? sqrt(x) : 0;
         bipolar = false;
         break;
case 4:  f = (x>0) ? pow(x,0.3333333) : 0;
         bipolar = false;
         break;
case 5:  f = (x>0) ? 1 : -1;
         bipolar = true;
         break;
case 6:  f = x / (10 * y);
         bipolar = true;
         break;
case 7:  f = sqrt(x*x+y*y);
         bipolar = false;
         break;
case 8:  f = pow(x,5*y);
         bipolar = false;
         break;
case 9:  f = exp(x-1);
         bipolar = false;
         break;
case 10: f = exp(2*x - 2);
         bipolar = false;
         break;
case 11: f = (x>0) ? 1 + log(x) : -1;
         bipolar = true;
         break;
case 12: f = (x>0) ? 1 + 0.5 * log(x) : -1;
         bipolar = true;
         break;
case 13: f = atan(y/x) / 3.141592; // -0.5 ... +0.5
         if (x<0)
           if (y>0) f += 1;        // +0.5 ... +1.0
           else f -= 1;            // -1.0 ... -0.5
         bipolar = true;
         break;
```

```
    case 14: f = sin(x * 3.141592);
             bipolar = true;
             break;
    case 15: f = cos(x * 3.141592);
             bipolar = true;
             break;
  }

  if (bipolar)
  {
    if (f < -1) f = -1;
    if (f > 1) f = 1;
    analogWrite(A0,(int)(511.5 + f * 511.5));   // bipolar output
  }
  else
  {
    if (f < 0) f = 0;
    if (f > 1) f = 1;
    analogWrite(A0,(int)(f * 1023));             // unipolar output
  }
}
```

This is an improved version of the software, using look-up-tables for all functions that depend only on one variable. These functions are now executed with about 50 kHz samplerate. Functions of two variables are slower:
x*y   29 kHz
x/(10y)   14.6 kHz
sqrt(x*x+y*y)   18.3 kHz
x^(5y)   4 kHz
atan2(y/x)/3.141592   4.7 kHz

```
/*
  Programmable Function Generator for THAT with Seeed_XIAO
  Version 2 with look-up-tables
  Michael Koch, January 2024
*/

bool bipolar;
int mode, lastMode;
unsigned int port;
float x,y,f;
int xx, yy;
float offset, gain;
union
{
  int i[4096];
  float f[4096];
} lut;

// the setup function runs once when you press reset or power the board
void setup()
{
  offset = -1.15;      // This can be fine-tuned
  gain = 0.00058;      // This can be fine-tuned
  lastMode = 255;
  pinMode(A1,INPUT);           // analog inputs
  pinMode(A2,INPUT);
  pinMode(D3, INPUT_PULLUP);   // pullup resistors for mode switch inputs
  pinMode(D4, INPUT_PULLUP);
  pinMode(D5, INPUT_PULLUP);
  pinMode(D6, INPUT_PULLUP);
  pinMode(D7, OUTPUT);         // Test pin for sample rate
  analogReadResolution(12);
  analogWriteResolution(10);
}
```

```
void fillLut(int mode)
{
  int i;
  float f;
  for (int x = 0; x < 4096; x++)
  {
    switch (mode)
    {
      case 0: f = offset + gain * x;
              if (f < -1) f = -1;
              if (f > 1) f = 1;
              lut.i[x] = (int)(2048 * f);
              break;
      case 1: i = (int)(1023 * sq(offset + gain * x));
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 2: i = (int)(511.5 + 511.5 * pow(offset + gain * x, 3));
              if (i < 0) i = 0;
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 3: f = offset + gain * x;
              f = (f>0) ? sqrt(f) : 0;
              i = (int)(1023 * f);
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 4: f = offset + gain * x;
              f = (f>0) ? pow(f, 0.33333) : 0;
              i = (int)(1023 * f);
              if (i > 1023) i = 1023;
              lut.i[x] = i;
              break;
      case 5: break;
      case 6: f = offset + gain * x;
              if (f < -1) f = -1;
              if (f > 1) f = 1;
              lut.f[x] = f;
              break;
      case 7: f = sq(offset + gain * x);
              if (f > 1) f = 1;
```

```
            lut.f[x] = f;
            break;
    case 8: f = offset + gain * x;
            if (f < -1) f = -1;
            if (f > 1) f = 1;
            lut.f[x] = f;
            break;
    case 9: f = offset + gain * x;
            f = exp(f - 1.0);
            i = (int)(1023 * f);
            if (i > 1023) i = 1023;
            lut.i[x] = i;
            break;
    case 10: f = offset + gain * x;
             f = exp(2 * f - 2.0);
             i = (int)(1023 * f);
             if (i > 1023) i = 1023;
             lut.i[x] = i;
             break;
    case 11: f = offset + gain * x;
             f = (f > -1) ? log(1 + f) : -1;
             if (f < -1) f = -1;
             if (f>1) f = 1;
             i = (int)(511.5 + 511.5 * f);
             lut.i[x] = i;
             break;
    case 12: f = offset + gain * x;
             f = (f > -1) ? 0.5 * log(1 + f) : -1;
             if (f < -1) f = -1;
             if (f>1) f = 1;
             i = (int)(511.5 + 511.5 * f);
             lut.i[x] = i;
             break;
    case 13: f = offset + gain * x;
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             lut.f[x] = f;
             break;
    case 14: f = offset + gain * x;
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             f = sin(f * 3.141592);
             lut.i[x] = (int)(511.5 + 511.5 * f);
```

```
              break;
      case 15: f = offset + gain * x;
              if (f < -1) f = -1;
              if (f > 1) f = 1;
              f = cos(f * 3.141592);
              lut.i[x] = (int)(511.5 + 511.5 * f);
              break;
    }
  }
}

// the loop function runs over and over again forever
void loop()
{
  // mode = 15 - digitalRead(D3);  // read the mode switch, this is slow
  // mode -= 2 * digitalRead(D4);
  // mode -= 4 * digitalRead(D5);
  // mode -= 8 * digitalRead(D6);
  mode = 15;
  port = PORT->Group[0].IN.reg;    // read the mode switch, this is faster
  if (port & 2048) mode -= 1;
  if (port & 256) mode -= 2;
  if (port & 512) mode -= 4;
  port = PORT->Group[1].IN.reg;
  if (port & 256) mode -= 8;

  if (mode != lastMode)         // if the mode has changed,
  {
    fillLut(mode);              // fill the look-up-table
    lastMode = mode;
  }

  // digitalWrite(D7, HIGH);                // test pin = high, this is slow
  PORT->Group[1].OUTSET.reg = 512;          // test pin = high, this is faster
  xx = analogRead(A1);    // read analog x input
  // digitalWrite(D7, LOW);                 // test pin = low, this is slow
  PORT->Group[1].OUTCLR.reg = 512;          // test pin = low, this is faster

  switch (mode)
  {
    case 0:  yy = analogRead(A2);        // x * y
             analogWrite(A0,511 + 512 * lut.i[xx] * lut.i[yy] / 4194304);
             break;
```

```
    case 1:  analogWrite(A0,lut.i[xx]);  // x^2
             break;
    case 2:  analogWrite(A0,lut.i[xx]);  // x^3
             break;
    case 3:  analogWrite(A0,lut.i[xx]);  // sqrt(x)
             break;
    case 4:  analogWrite(A0,lut.i[xx]);  // x^(1/3)
             break;
    case 5:  if (xx < 2048)              // sgn(x)
               analogWrite(A0,0);
             else
               analogWrite(A0,1023);
             break;
    case 6:  yy = analogRead(A2);        // x / (10 * y)
             f = lut.f[xx] / (10 * lut.f[yy]);
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             analogWrite(A0,(int)(511.5 + f * 511.5));
             break;
    case 7:  yy = analogRead(A2);        // sqrt(x*x+y*y)
             f = sqrt(lut.f[xx] + lut.f[yy]);
             if (f > 1) f = 1;
             analogWrite(A0,(int)(f * 1023));
             break;
    case 8:  yy = analogRead(A2);        // x^(5*y)
             f = (xx > 2047) ? pow(lut.f[xx], 5 * lut.f[yy]) : 0;
             if (f > 1) f = 1;
             analogWrite(A0,(int)(f * 1023));
             break;
    case 9:  analogWrite(A0,lut.i[xx]);  // exp(x-1)
             break;
    case 10: analogWrite(A0,lut.i[xx]);  // exp(2x-2)
             break;
    case 11: analogWrite(A0,lut.i[xx]);  // log(1+x)
             break;
    case 12: analogWrite(A0,lut.i[xx]);  // 0.5 * log(1+x)
             break;
    case 13: yy = analogRead(A2);        // atan2(y/x)/3.141592
             x = lut.f[xx];
             y = lut.f[yy];
             f = atan(y/x) / 3.141592; // -0.5 ... +0.5
             if (x<0)
               if (y>0) f += 1;        // +0.5 ... +1.0
```

```
              else f -= 1;            // -1.0 ... -0.5
             if (f < -1) f = -1;
             if (f > 1) f = 1;
             analogWrite(A0,(int)(511.5 + f * 511.5));
             break;
    case 14: analogWrite(A0,lut.i[xx]);  // sin(x * 3.141592)
             break;
    case 15: analogWrite(A0,lut.i[xx]);  // cos(x * 3.141592)
             break;
  }
}
```

More info about Seeed XIAO SAMD21:

https://wiki.seeedstudio.com/SeeedStudio_XIAO_Series_Introduction/

https://wiki.seeedstudio.com/Seeeduino-XIAO/

SAMD21 Data sheet: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/ATSAMD21G18A-MU-Datasheet.pdf

Schematic diagram: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/Seeeduino-XIAO-v1.0-SCH-191112.pdf

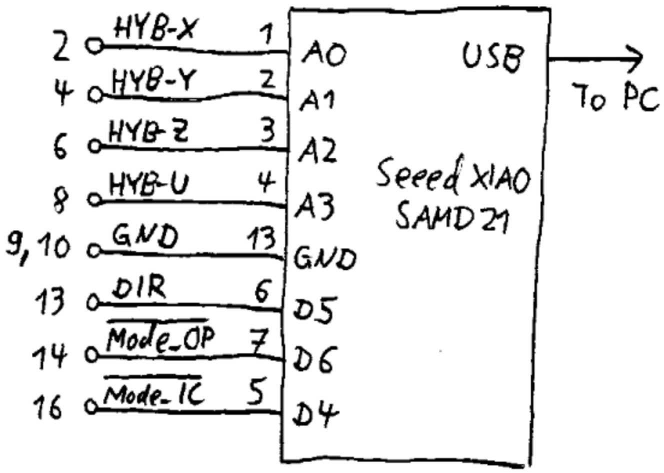
Wearable Projects Step by Step: https://files.seeedstudio.com/wiki/Seeeduino-XIAO/res/Seeeduino-XIAO-in-Action-Minitype%EF%BC%86Wearable-Projects-Step-by-Step.pdf

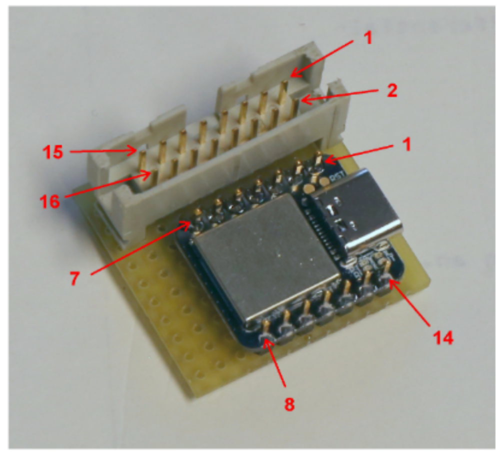
The ADC is 12-bit and the DAC is 10-bit.

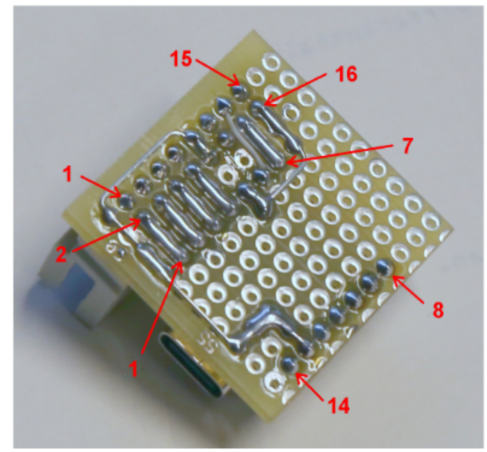
The clock frequency is 48 MHz, derived via PLL from a 32.768 kHz crystal.

4.4 Hybrid Port Oscilloscope with Seeed XIAO SAMD21

The schematic diagram is very simple. If you need a 4-channel oscilloscope for THAT which can be built for less than 30 EUR, here it is!
The pin numbers at the left side are for THATs hybrid port:







It's important that pins 11 and 12 of THATs hybrid port aren't connected to any pins of the microcontroller, because these are +5V pins and the microcontroller's input and output pins aren't 5V tolerant. The microcontroller gets its own supply voltage over the USB cable.

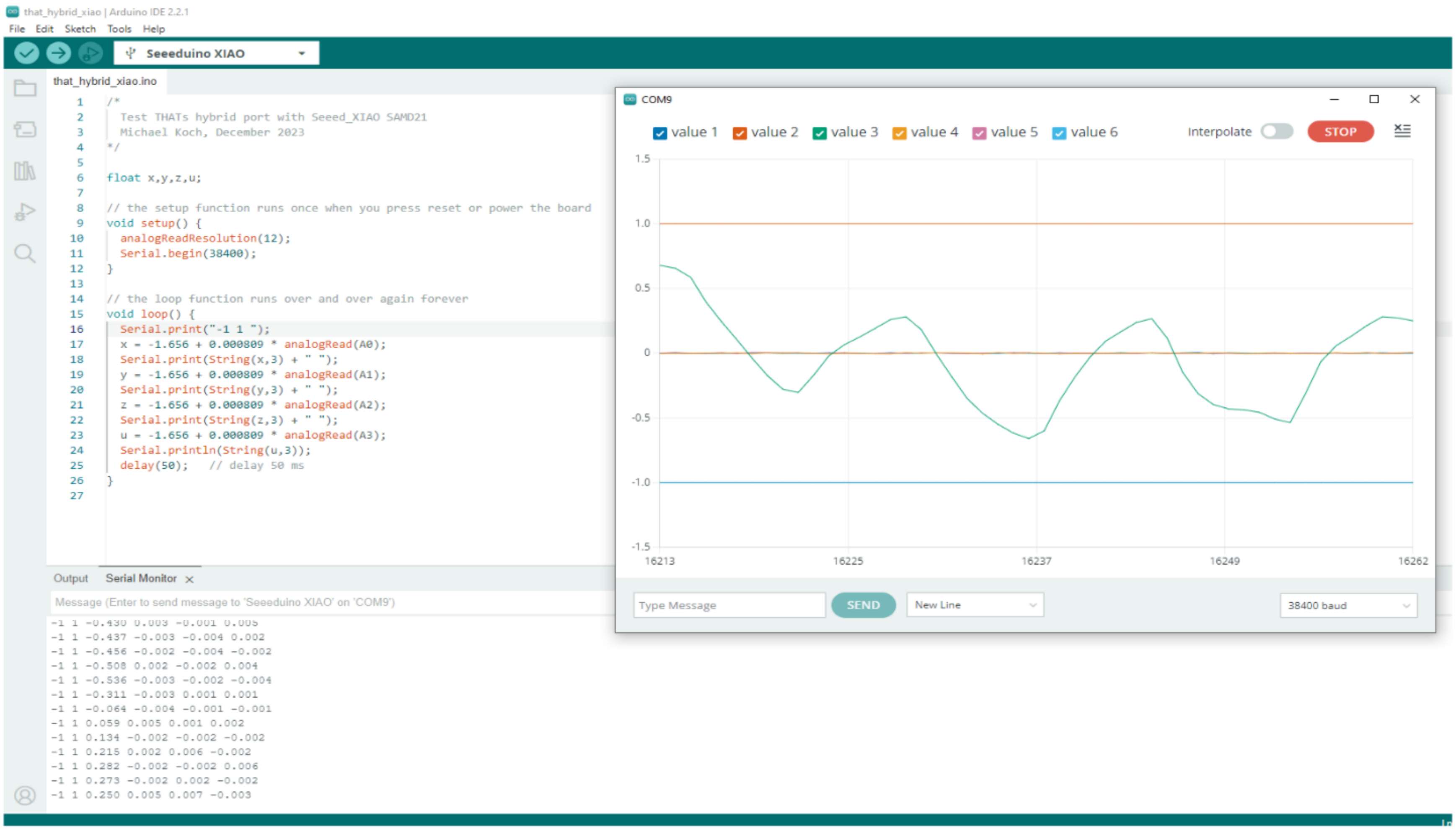
This is a very simple (preliminary) software for testing the hybrid port oscilloscope. It's much easier than I thought at first, because the Arduino IDE has already a built-in serial plotter. The software just reads four analog inputs, converts the values to machine units and writes the results to the serial port. The sampling speed can be changed in the last line of the source code.

```
/*
  Test THATs hybrid port with Seeed_XIAO SAMD21
  Michael Koch, December 2023
*/

float x,y,z,u;

// the setup function runs once when you press reset or power the board
void setup() {
  analogReadResolution(12);
}

// the loop function runs over and over again forever
void loop() {
  Serial.print("-1 1 ");
  x = -1.656 + 0.000809 * analogRead(A0);
  Serial.print(String(x,3) + " ");
  y = -1.656 + 0.000809 * analogRead(A1);
  Serial.print(String(y,3) + " ");
  z = -1.656 + 0.000809 * analogRead(A2);
  Serial.print(String(z,3) + " ");
  u = -1.656 + 0.000809 * analogRead(A3);
  Serial.println(String(u,3));
  delay(50);   // delay 50 ms
}
```

The constants for offset and gain in lines 17, 19, 21 and 23 may need some fine tuning, because the microcontroller uses its own 3.3V supply as analog reference.

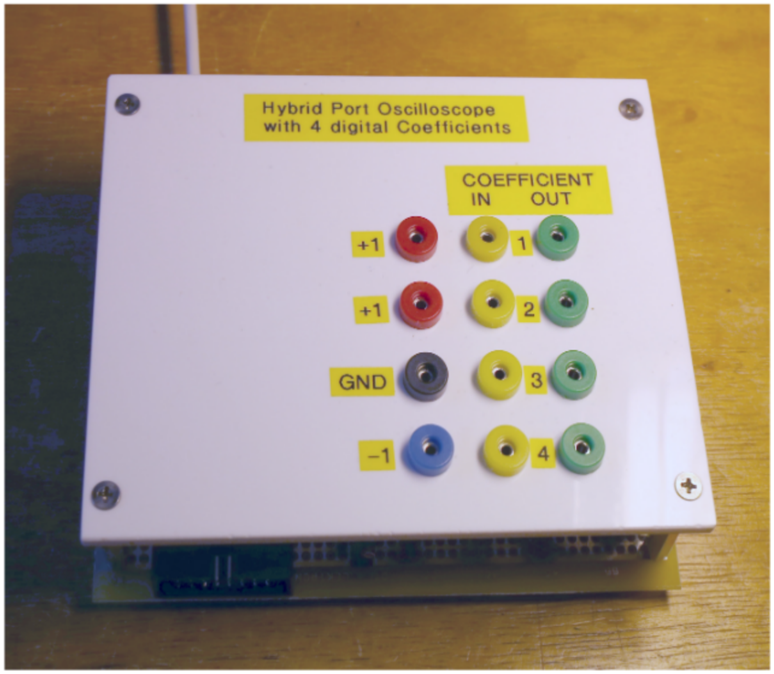
For a better software with much more features, see next chapter.

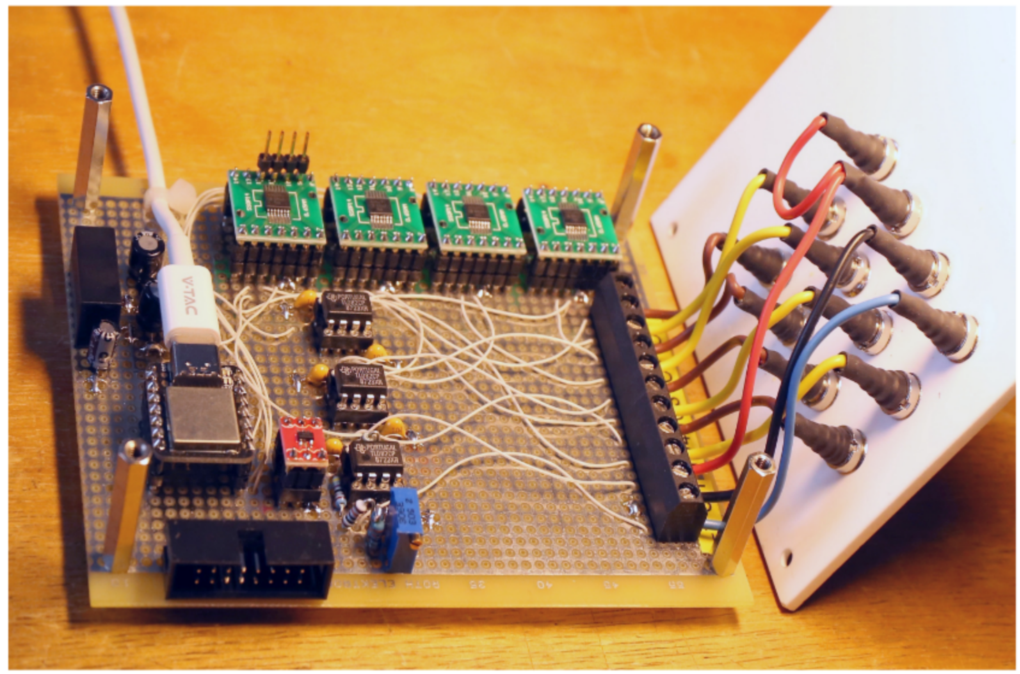
At the top left of the screenshot you see the source code. At the bottom left you see the serial output. At the right is the serial plotter.
Why did I also write two constants -1 and 1 ? That's because the serial plotter does automatically scale the vertical axis to the minimum and maximum in the data, which I found quite disturbing. Writing -1 and 1 effectively disables this autoscale feature.



4.5 Hybrid Port Oscilloscope with 4 digital Coefficients

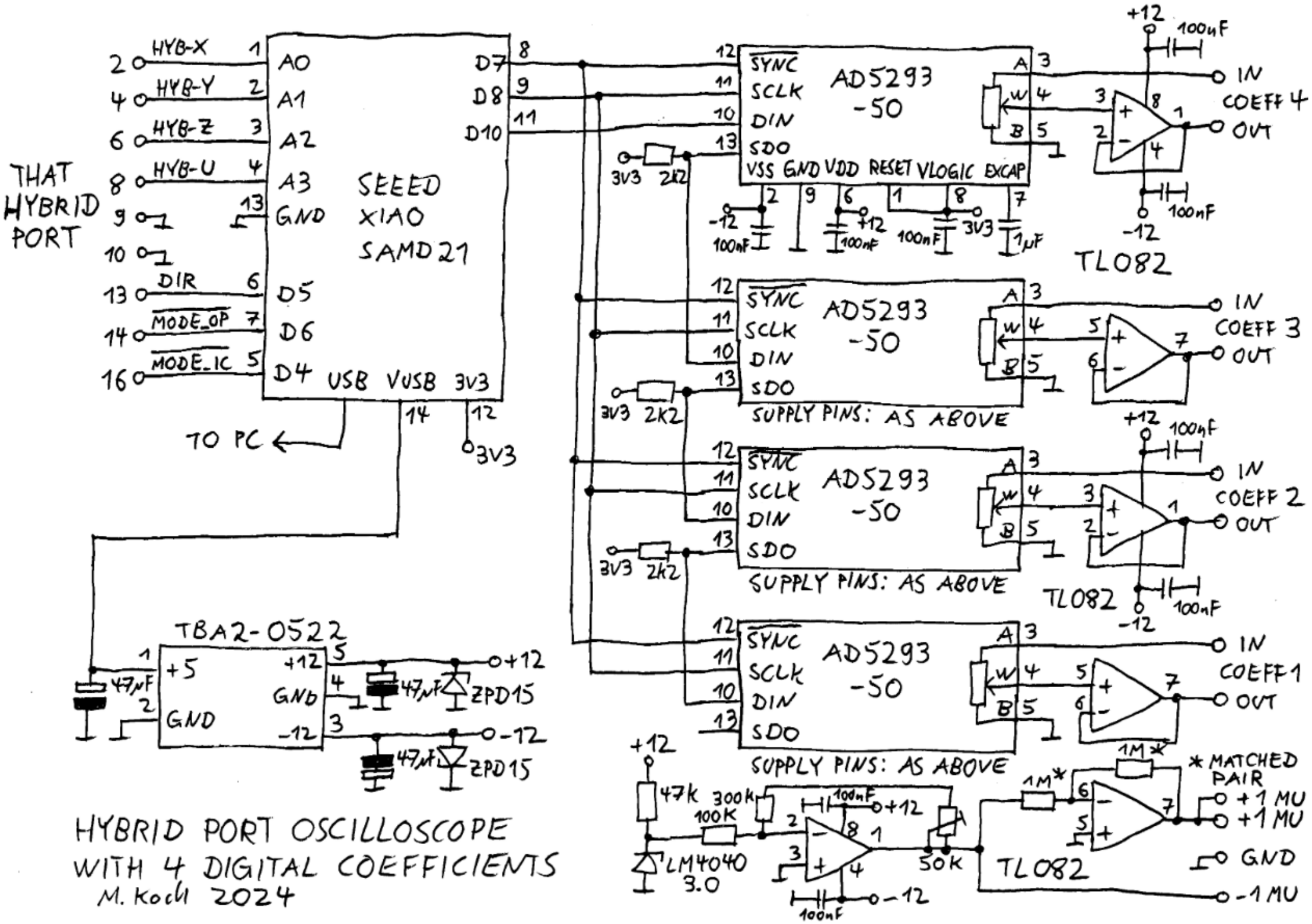
This is the same hybrid port as above, but with four digital coefficients added.





2mm jacks from Aliexpress: https://de.aliexpress.com/item/4000271665708.html

Prototype board with groundplane: https://www.reichelt.de/hf-europlatine-epoxyd-verzinnt-160x100mm-re-201lf-p34772.html


THAT HYBRID PORT
HYB-X
HYB-Y
HYB-Z
HYB-U
DIR
MODE_OP
MODE_IC
SEEED XIAO SAMD21
TO PC
3V3
2k2
AD5293 -50
SYNC
SCLK
DIN
SDO
VSS GND VDD RESET VLOGIC EXCAP
SUPPLY PINS: AS ABOVE
TL082
IN
COEFF 4
OUT
COEFF 3
COEFF 2
COEFF 1
100nF
1µF
+12
-12
TBA2-0522
47µF
ZPD15
GND
47k
300k
100k
LM4040 3.0
50k
1M*
* MATCHED PAIR
+1 MU
-1 MU
HYBRID PORT OSCILLOSCOPE WITH 4 DIGITAL COEFFICIENTS
M. Koch 2024

Below is the listing of the software with many features. The same software can also be used for the previous circuit (without digital coefficients):

- **Samplerate from 1Hz to 40kHz (20kHz with 2 channels, 10kHz with 4 channels)**
- **1, 2 or 4 channels**
- **50 to 2000 samples per channel**
- **Oversampling, the ADCs are always running with 40kHz (20kHz with 2 channels, 10kHz with 4 channels)**
- **Full remote control of THAT: IC, OP, HALT**
- **Controlled via serial port from Arduino IDE**
- **Supports the "Serial plotter" in Arduino IDE (which shows only 50 samples per channel, but this number can be increased by a dirty hack)**
- **Built-in help function lists all commands, press "?"**

```
/*
  Oscilloscope for THATs hybrid port with Seeed_XIAO SAMD21
  with 4 digital potentiometers
  Michael Koch, January 2024
*/

#include <TimerTC3.h>
#include <SPI.h>

int channels, samplerate, samples, oversampling;
int ch, ch1, ch2, ch3, ch4, samplesToAcquire;
int ovs1, ovs2, ovs3, ovs4;
int coeff[4];
bool hybrid, serialPlotter, labels;
unsigned short waveform[8000];
int writePointer;

// the setup function runs once when you press reset or power the board
void setup()
{
  pinMode(D7,OUTPUT);
  digitalWrite(D7,HIGH);   // CS = high
  pinMode(D8,OUTPUT);
  pinMode(D10,OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  analogReadResolution(12);
  Serial.setTimeout(4294967295);   // about 50 days
```

```
  channels = 4;      // initial settings
  samples = 500;
  samplesToAcquire = 0;
  samplerate = 5000;
  hybrid = 1;
  serialPlotter = 1;
  labels = 1;
  oversampling = 40000 / samplerate / channels;

  coeff[0] = coeff[1] = coeff[2] = coeff[3] = 0;

  if (hybrid)
  {
    pinMode(D5,OUTPUT);
    digitalWrite(D5,LOW);   // DIR = low
    pinMode(D4,OUTPUT);
    digitalWrite(D4,HIGH);   // Mode_IC = high
    pinMode(D6,OUTPUT);
    digitalWrite(D6,HIGH);   // Mode_OP = high
  }

  SPI.begin ();
  SPI.setClockDivider(SPI_CLOCK_DIV2);  // divide the clock by 2
  SPI.setDataMode(SPI_MODE1);

  digitalWrite(D7,LOW);   // CS = low   // disable the write protection
  SPI.transfer16(0x1802);               //  in all four AD5293
  SPI.transfer16(0x1802);
  SPI.transfer16(0x1802);
  SPI.transfer16(0x1802);
  digitalWrite(D7,HIGH);  // CS = high
  delayMicroseconds(1);
  digitalWrite(D7,LOW);   // CS = low   // set all four AD5293 to zero
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  SPI.transfer16(0x0400);
  digitalWrite(D7,HIGH);  // CS = high

  TimerTc3.initialize(25);   // 25 us, 40kHz
  TimerTc3.attachInterrupt(timerIsr);
}

void timerIsr()   // 40kHz timer interrupt service routine
{
```

```
  if (samplesToAcquire == 0)
    digitalWrite(LED_BUILTIN,HIGH);   // orange LED off
  else
  {
    switch (ch)
    {
      case 0: ch1 += analogRead(A0);
              if(--ovs1 == 0)
              {
                ovs1 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch1 / oversampling);
                ch1 = 0;
              }
              break;
      case 1: ch2 += analogRead(A1);
              if(--ovs2 == 0)
              {
                ovs2 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch2 / oversampling);
                ch2 = 0;
              }
              break;
      case 2: ch3 += analogRead(A2);
              if(--ovs3 == 0)
              {
                ovs3 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch3 / oversampling);
                ch3 = 0;
              }
              break;
      case 3: ch4 += analogRead(A3);
              if(--ovs4 == 0)
              {
                ovs4 = oversampling;
                waveform[writePointer++] = (unsigned short)(ch4 / oversampling);
                ch4 = 0;
              }
              break;
    }
    samplesToAcquire -= 1;
    ch = (ch+1) % channels;   // 1 channel: ch=0    2 channels: ch=0,1    4 channels: ch=0,1,2,3
  }
}

void acquire()
{
```

```
writePointer = 0;    // set the initial conditions for a new acquisition
ovs1 = ovs2 = ovs3 = ovs4 = oversampling;
ch1 = ch2 = ch3 = ch4 = 0;
ch = 0;

if (hybrid)
{
  digitalWrite(D4,LOW);      // Mode_IC = low
  digitalWrite(D6,HIGH);     // Mode_OP = high
  delay(10);                 // wait 10ms
  digitalWrite(D4,HIGH);     // Mode_IC = high
  digitalWrite(D6,LOW);      // Mode_OP = low
}
digitalWrite(LED_BUILTIN,LOW);   // orange LED on
samplesToAcquire = samples * oversampling * channels;   // this starts the acquisition

for( ; samplesToAcquire != 0; )   // wait for end of acquisition
  delay(1);

int index = 0;
for (int n=0; n < samples; n++)   // print the waveforms
{
  for (int j = 0; j < channels; j++)
  {
    if (labels)
    {
      switch (j)
      {
        case 0: Serial.print("X:");
                break;
        case 1: Serial.print("Y:");
                break;
        case 2: Serial.print("Z:");
                break;
        case 3: Serial.print("U:");
                break;
      }
    }
    Serial.print(String(-1.656 + 0.000809 * waveform[index++],3) + " "); // offet and gain constants can be fine-tuned
  }
  if(serialPlotter)
    if (labels)
      Serial.print("MIN:-1 MAX:1");
    else
      Serial.print("-1 1");
  Serial.println();
```

```
  }
}

// the loop function runs over and over again forever
void loop()
{
  String arg1, arg2, arg3;

  String s = Serial.readStringUntil('\n');
  s.toUpperCase();
  s.trim();
  int i = s.indexOf(' ');
  arg1 = s.substring(0,i);
  if (i!=-1)
    arg2 = s.substring(i);
  else
    arg2 = "";
  arg1.trim();
  arg2.trim();

  if (arg1 == "")       // Acquire
    acquire();

  if (arg1 == "L")      // List settings
  {
    Serial.print(" Channels: ");
    Serial.println(channels);
    Serial.print(" Samples per channel: ");
    Serial.println(samples);
    Serial.print(" Samplerate: ");
    Serial.print(samplerate);
    Serial.println(" Hz");
    Serial.print(" Oversampling factor: ");
    Serial.println(oversampling);
    Serial.print(" Coefficient 1: ");
    Serial.println((float)coeff[0] / 1024.0, 3);
    Serial.print(" Coefficient 2: ");
    Serial.println((float)coeff[1] / 1024.0, 3);
    Serial.print(" Coefficient 3: ");
    Serial.println((float)coeff[2] / 1024.0, 3);
    Serial.print(" Coefficient 4: ");
    Serial.println((float)coeff[3] / 1024.0, 3);
    Serial.print(" Hybrid mode: ");
    Serial.println(hybrid ? "on" : "off");
    Serial.print(" Arduino serial plotter mode: ");
    Serial.println(serialPlotter ? "on" : "off");
```

```
  Serial.print(" Label mode: ");
  Serial.println(labels ? "on" : "off");
  Serial.println();
}

if (arg1 == "CH")     // Set number of channels
{
  switch (arg2.toInt())
  {
    case 1: channels = 1;
            Serial.println("1 Channel");
            break;
    case 2: if (samplerate < 40000)
            {
              channels = 2;
              Serial.println("2 Channels");
            }
            else
            {
              Serial.println("Error: Samplerate is too high");
            }
            break;
    case 4: if (samplerate < 20000)
            {
              channels = 4;
              Serial.println("4 Channels");
            }
            else
            {
              Serial.println("Error: Samplerate is too high");
            }
            break;
    default: Serial.println("Error: Number of channels must be 1, 2 or 4");
  }
  oversampling = 40000 / samplerate / channels;
}

if (arg1 == "HY")     // Set hybrid mode on / off
{
  switch (arg2.toInt())
  {
    case 0: hybrid = 0;
            pinMode(D4,INPUT);
            pinMode(D5,INPUT);
            pinMode(D6,INPUT);
            Serial.println("Hybrid mode off");
```

```
            break;
    case 1: hybrid = 1;
            pinMode(D5,OUTPUT);
            digitalWrite(D5,LOW);   // DIR = low
            pinMode(D4,OUTPUT);
            digitalWrite(D4,HIGH);   // Mode_IC = high
            pinMode(D6,OUTPUT);
            digitalWrite(D6,HIGH);   // Mode_OP = high
            Serial.println("Hybrid mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}

if (arg1 == "SP")     // Set serial plotter mode on / off
{
  switch (arg2.toInt())
  {
    case 0: serialPlotter = 0;
            Serial.println("Serial plotter mode off");
            break;
    case 1: serialPlotter = 1;
            Serial.println("Serial plotter mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}

if (arg1 == "LA")     // Set label mode on / off
{
  switch (arg2.toInt())
  {
    case 0: labels = 0;
            Serial.println("Label mode off");
            break;
    case 1: labels = 1;
            Serial.println("Label mode on");
            break;
    default: Serial.println("Error: Argument must be 0 or 1");
  }
}

if (arg1 == "IC")     // Set THAT to IC mode
{
  if (hybrid)
  {
```

```
    digitalWrite(D4,LOW);     // Mode_IC = low
    digitalWrite(D6,HIGH);    // Mode_OP = high
    delay(10);
    Serial.println("THAT set to IC mode");
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "OP")     // Set THAT to OP mode
{
  if (hybrid)
  {
    digitalWrite(D4,HIGH);    // Mode_IC = high
    digitalWrite(D6,LOW);     // Mode_OP = low
    Serial.println("THAT set to OP mode");
    acquire();
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "HA")     // Set THAT to HALT mode
{
  if (hybrid)
  {
    digitalWrite(D4,HIGH);    // Mode_IC = high
    digitalWrite(D6,HIGH);    // Mode_OP = high
    Serial.println("THAT set to HALT mode");
  }
  else
    Serial.println("Error: THAT is not in hybrid mode");
}

if (arg1 == "SA")     // Set number of samples
{
  int c = arg2.toInt();
  if ((c >= 50) && (c <= 2000))
  {
    samples = c;
    Serial.print(samples);
    Serial.println(" Samples");
  }
  else
    Serial.println("Error: Number of samples must be between 50 and 2000");
}
```

```
if (arg1 == "CO")      // Set coefficient
{
  int i = arg2.indexOf(' ');
  arg3 = arg2.substring(i);
  arg2 = arg2.substring(0,i);
  arg2.trim();
  arg3.trim();
  int co_nr = arg2.toInt() - 1;
  if ((co_nr < 0) || (co_nr > 3))
    Serial.println("Error: Coefficient number must be 1, 2, 3 or 4");
  else
  {
    int c = (int)(1024 * arg3.toFloat());
    if (c < 0) c = 0;
    if (c > 1023) c = 1023;
    coeff[co_nr] = c;
    digitalWrite(D7,LOW);   // CS = low
    SPI.transfer16(0x0400 + coeff[0]);   // write data to all 4 AD5293
    SPI.transfer16(0x0400 + coeff[1]);
    SPI.transfer16(0x0400 + coeff[2]);
    SPI.transfer16(0x0400 + coeff[3]);
    digitalWrite(D7,HIGH);   // CS = high
    Serial.print(" Coefficient ");
    Serial.print(co_nr + 1);
    Serial.print(" was set to ");
    Serial.println((float)c / 1024.0);
  }
}

if (arg1 == "SR")      // Set samplerate
{
  int c = arg2.toInt();
  switch (c)
  {
    case 1:
    case 2:
    case 5:
    case 10:
    case 20:
    case 50:
    case 100:
    case 200:
    case 500:
    case 1000:
    case 2000:
```

```
    case 5000:
    case 10000: samplerate = c;
                break;
    case 20000: if (channels != 4)
                  samplerate = c;
                else
                  Serial.println("Error: Samplerate 20000 isn't supported with 4 channels");
                break;
    case 40000: if (channels == 1)
                  samplerate = c;
                else
                  Serial.println("Error: Samplerate 40000 isn't supported with 2 or 4 channels");
                break;
    default: Serial.println("Error: Invalid samplerate");
  }
  oversampling = 40000 / samplerate / channels;
  Serial.print(" Samplerate: ");
  Serial.print(samplerate);
  Serial.println(" Hz");
}

if (arg1 == "?")
{
  Serial.println("****************************************************************************");
  Serial.println("THAT Hybrid Port Oscilloscope with Seeed XIAO SAMD21   (C) Michael Koch 2024");
  Serial.println("****************************************************************************");
  Serial.println("");
  Serial.println("List of possible samplerates and oversampling factors:");
  Serial.println("Samplerate  Sample  Oversampling Factor");
  Serial.println("     Hz     Interval   1Ch    2Ch    4Ch ");
  Serial.println("   40000     25us       1      -      - ");
  Serial.println("   20000     50us       2      1      - ");
  Serial.println("   10000    100us       4      2      1 ");
  Serial.println("    5000    200us       8      4      2 ");
  Serial.println("    2000    500us      20     10      5 ");
  Serial.println("    1000      1ms      40     20     10 ");
  Serial.println("     500      2ms      80     40     20 ");
  Serial.println("     200      5ms     200    100     50 ");
  Serial.println("     100     10ms     400    200    100 ");
  Serial.println("      50     20ms     800    400    200 ");
  Serial.println("      20     50ms    2000   1000    500 ");
  Serial.println("      10    100ms    4000   2000   1000 ");
  Serial.println("       5    200ms    8000   4000   2000 ");
  Serial.println("       2    500ms   20000  10000   5000 ");
  Serial.println("       1      1s    40000  20000  10000 ");
  Serial.println("");
```

```
    Serial.println("List of commands:");
    Serial.println("?       This help page");
    Serial.println("IC      Set THAT to IC mode and wait 10ms");
    Serial.println("         This works only if THAT is in hybrid mode, see HY command");
    Serial.println("OP      Set THAT to OP mode, and start data acquisition");
    Serial.println("         This works only if THAT is in hybrid mode, see HY command");
    Serial.println("HA      Set THAT to HALT mode");
    Serial.println("         This works only if THAT is in hybrid mode, see HY command");
    Serial.println("<cr>    If THAT is in hybrid mode: Set THAT to IC, wait 10ms, then set THAT to OP and start data acquisition");
    Serial.println("        If THAT is not in hybrid mode: Start data acquisition");
    Serial.println("CH n    Set the number of channels, n = 1, 2 or 4, default is 4");
    Serial.println("         2 channels are only possible with samplerate 20000 or slower");
    Serial.println("         4 channels are only possible with samplerate 10000 or slower");
    Serial.println("SA n    Set the number of samples per channel, n = 50 to 2000, default is 500");
    Serial.println("         By default the Arduino IDE's serial plotter shows only 50 samples per channel");
    Serial.println("         But it's possible to change it to 2000 with a dirty hack");
    Serial.println("SR n    Set samplerate in Hz, n = 1 to 40000 (see table above), default is 1000");
    Serial.println("         Samplerate 20000 is only possible with 1 or 2 channels");
    Serial.println("         Samplerate 40000 is only possible with 1 channel");
    Serial.println("         The internal samplerate is always as high as possible for oversampling");
    Serial.println("CO n x  Set coefficient n (1,2,3,4) to the value x (0.000 to 1.000)");
    Serial.println("          The resolutiuon is 10-bit or about 0.001");
    Serial.println("HY b    Set THAT to hybrid mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         0 means THAT is controlled by its MODE switch");
    Serial.println("         1 means THAT is controlled via hybrid port");
    Serial.println("SP b    Select serial plotter mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         1 means two dummy channels with constants -1 and 1 are added,");
    Serial.println("         so that the serial plotter's autoscale feature is disabled");
    Serial.println("LA b    Label mode, can be 0 (off) or 1 (on), default is 1");
    Serial.println("         1 means the signals are labeled X, Y, Z, U, MIN and MAX");
    Serial.println("         0 means no labels are sent so the serial plotter");
    Serial.println("L       List the actual settings");
    Serial.println("");
  }
}
```

4.6 Software for serial plotting

The "Serial plotter" in Arduino IDE V2.2.1 displays only 50 points on the x axis by default.

The older version V1.8.18 did show 500 points, but had a [-5 ... +5] or [-6 ... +6] range for the y axis.

Workaround for showing more than 50 samples in the serial plotter in Arduino IDE V2.2.1:

This file must be hacked:

C:\Program Files\Arduino IDE\resources\app\lib\backend\resources\arduino-serial-plotter-webapp\static\js\main.35ae02cb.chunk.js

Before you can edit this file, you must change the security properties of the file, and allow write access by the user.

Then open the file in Notepad, search for

U=Object(o.useState)(50)

and replace it by

U=Object(o.useState)(2000)   (or use any other number of samples)

Save the file.

Now restart the Arduino IDE and the serial plotter will show 2000 samples!

Most probably this hack must be repeated when the Arduino IDE is updated to a new version.

I found this hack here (thank you Alex Tsyganok!), but on my computer the path was different than he described:

https://github.com/arduino/arduino-ide/issues/803#issuecomment-1338149431

Better Serial Plotter (freeware), unfortunately it does only receive data and it's not possible to send commands to the microcontroller:
https://hackaday.io/project/181686-better-serial-plotter

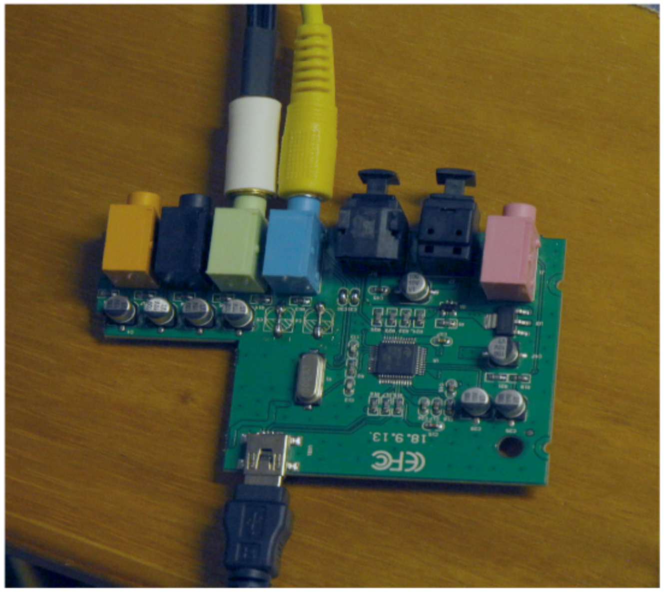
This "Processing Grapher" project might also be useful: https://wired.chillibasket.com/processing-grapher/

4.7 FFplay with USB Soundcard Input / Output

FFplay can be used for reading a signal from an audio input, processing the signal with an arbitrary transfer function, and sending the result to the computer's soundcard output. I did test it with a cheap USB soundcard. The audio inputs and outputs are AC coupled, but the capacitors can be removed and replaced by wires. Now the soundcard is DC coupled. The signal flow is as follows: A analog sawtooth signal goes into the soundcard input, it's converted to 16-bit with 8kHz sample rate, sent over USB to the PC, where the transfer function is applied in realtime, and then the signal is sent over the same USB cable back to the soundcard and converted to a voltage. I wanted to know the delay time from input to output. Unfortunately it's 80ms. Too slow for a realtime transfer function. This is the command line:

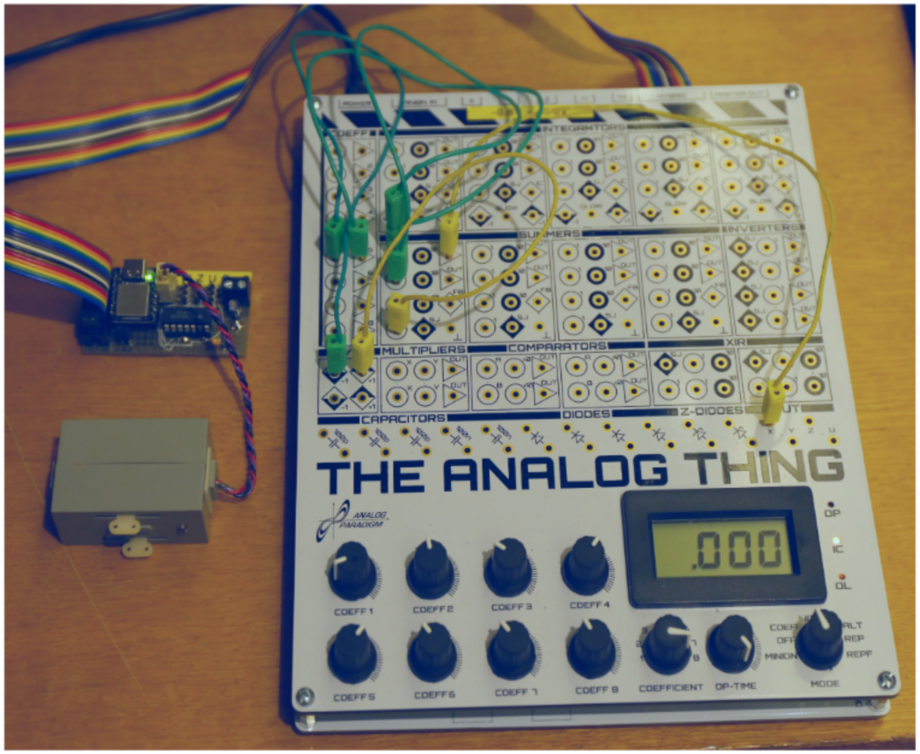
```
ffplay -nodisp -f dshow -audio_buffer_size 5 -sample_rate 8000 -sample_size 16 -channels 1 -i audio="Line (USB Sound
Device )" -af aeval='if(between(val(0),0,0.2),0,val(0))'
```

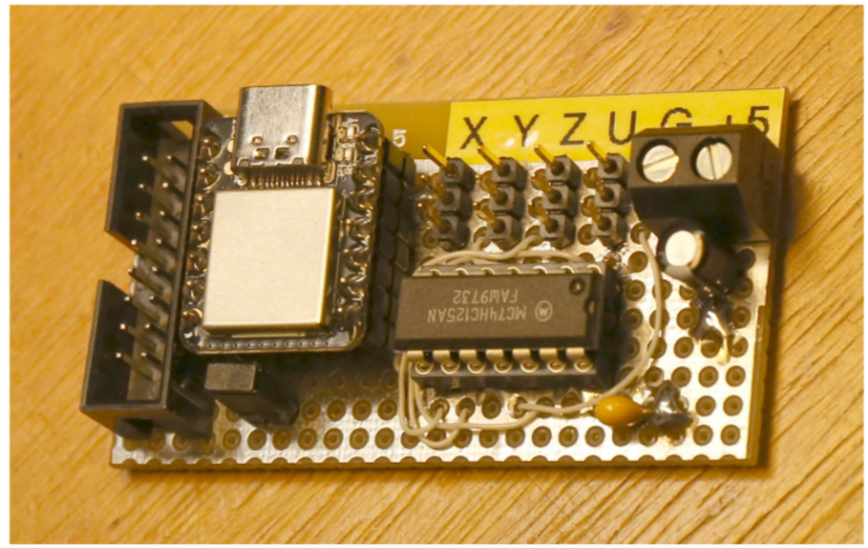
The last part of the command line is the transfer function. In words: If the input signal is between 0 and 0.2, then the output is 0, otherwise the output is the same as the input. On the oscilloscope magenta is input and yellow is output.



4.8 Controlling Servos with the Hybrid Port

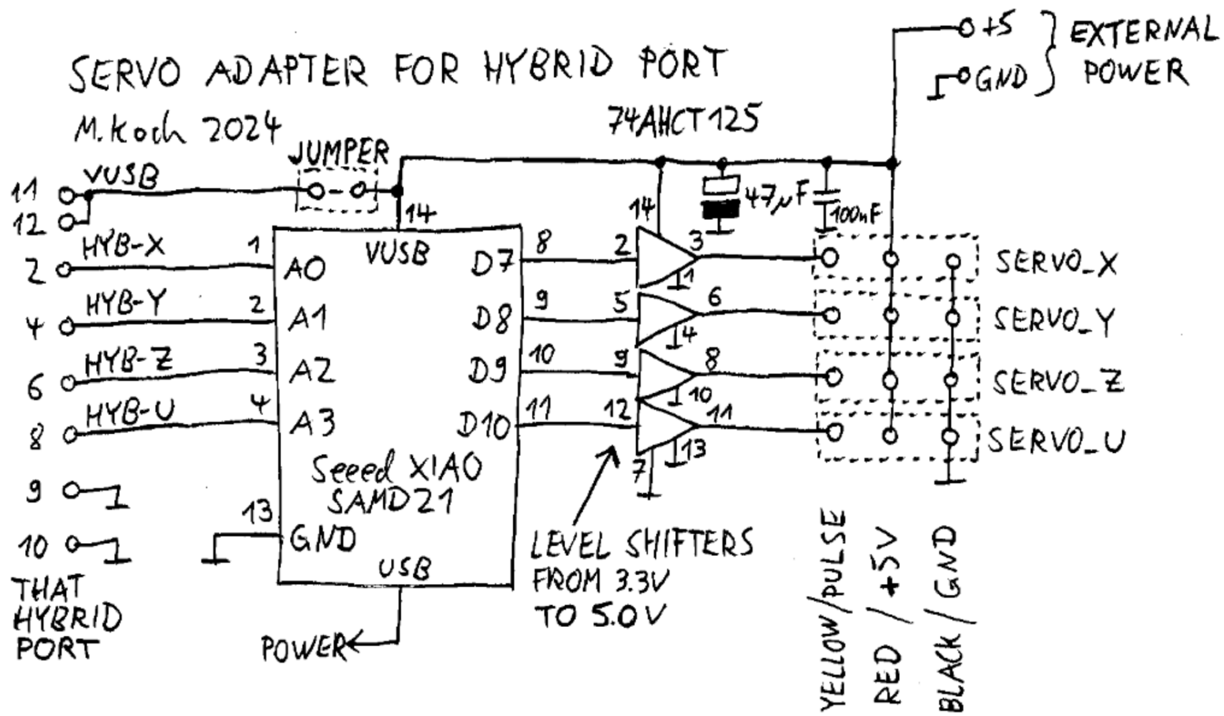
This is an adapter for controlling four servos from the hybrid port. The four servo's positions are proportional to the X, Y, Z and U signals.





Power for the servos can come from three sources:

- From THATs hybrid port, in this case install the jumper and don't connect a USB cable to the SAMD21 controller, and don't connect external power.
  This is only suitable for small servos, because THAT has an internal 750mA PTC fuse.
- From SAMD21 USB port, in this case remove the jumper and don't connect external power.
  This is only suitable for small servos, because the USB port can only supply 1000mA.
- From external power connector, in this case remove the jumper and don't connect a USB cable to the SAMD21 controller.
  This is also suitable for large servos. The external power connector can also be used for servos which need only 4.2V supply voltage.



See also: https://www.arduino.cc/reference/en/libraries/servo/

Cheap servos: https://de.aliexpress.com/item/1005005857192248.html

Software:

```
/*
  THAT Hybrid Port Adapter for 4 Servos
  Michael Koch, January 2024
*/

#include <Servo.h>

Servo myservo1, myservo2, myservo3, myservo4;
int minimum, maximum, mean, ampl;
float gain, offset;

// the setup function runs once when you press reset or power the board
void setup()
{
  analogReadResolution(12);
  minimum = 500;            // minimum pulse width in us
  maximum = 2500;           // maximum pulse width in us
  mean = (minimum + maximum) / 2;
  ampl = maximum - mean;
  gain = 0.000809;          // gain
  offset = -1.656;          // offset
  myservo1.attach(D7,minimum,maximum);
  myservo2.attach(D8,minimum,maximum);
  myservo3.attach(D9,minimum,maximum);
  myservo4.attach(D10,minimum,maximum);
}

// the loop function runs over and over again forever
void loop()
{
  float f;
  f = offset + gain * analogRead(A0);                   // -1 to +1
  myservo1.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A1);                   // -1 to +1
  myservo2.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A2);                   // -1 to +1
  myservo3.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  f = offset + gain * analogRead(A3);                   // -1 to +1
  myservo4.writeMicroseconds(mean + (int)(ampl * f));  // min to max
  delay(10);
}
```

List of servos (sorted by speed):

Servo Type	Voltage [V]	Current [A]	Torque [kg*cm]	Speed [s/60°]	Weight [g]	Notes and Links
Align DS825	7.4-8,4		10 @7.4V 12.5 @8.4V	0.030 @7.4V 0.020 @8.4V	62 or 72?	https://www.flashrc.com/de/servos/24744-align-ds825-brushless-hv-digital-servo-hsd82502-62g-12-5kg-cm-0-02s-60.html https://www.freakware.de/p/ds825m-high-voltage-brushless-servo-align-hsd82501-a147730.htm
ST4015	-8.4		15.5	0.020 @8.4V	75	https://www.modell-hubschrauber.at/Elektronisches-Zubehoer/Servos/Servos/Heckservos/Standard-Heckservos/1st-Brushless-Heckservo-ST-4015MG-HV-Digital::42675.html
Savöx SB-2272MG+	4.8-8.4		7 @6.0V 10 @8.4V	0.032 @6V 0.028 @8.4V	69	https://www.savox-shop.com/de/savoex-sb-2272mg-servo.html
DS835M	7.0-8.4		10.0 @7.4V 12.5 @8.4V	0.030 @7.4V 0.028 @8.4V	72	https://www.freakware.de/p/ds835m-high-voltage-brushless-servo-align-hsd83501-a177453.htm
BL855H	6.0-8.4		8 @6.0V 10 @7.4V 12.5 @8.4V	0.040 @6.0V 0.030 @7.4V 0.028 @8.4V	73	https://www.freakware.de/p/bl855h-high-voltage-brushless-servo-align-hsl85501-a106191.htm
Yantrs 0318AS-MGX-E	6.0-8.4	6.0	12 @4.2A 16 @5.5A 18 @6.0A	0.040 @6.0V 0.035 @7.4V 0.030 @8.4V	70	https://yantrs.cn/products/yantrs-helical-0-04s-27kg-low-profile-rc-servo-all-metal-waterproof-brushless-servo-%E7%9A%84%E5%89%AF%E6%9C%AC
DS835	6.0-8.4		5.0 @6.0V 6.5 @8.4V	0.050 @6.0V 0.035 @8.4V	29	https://www.freakware.de/p/ds535-digital-servo-align-hsd53502-a146314.htm
BH9012	6.0-8.4	3.3	8.5 @6.0V 9.5 @7.4V 12.0 @8.4V	0.050 @6.0V 0.040 @7.4V 0.035 @8.4V	70	https://moduapp.de/shop/shop/srt-brushless-servo-hv-high-speed-9-5kg-0-04sec-7-4v/
HiTEC HSG-8315BH	6.0-7.4		4.3 @6.0V 5.3 @7.4V	0.050 @6.0V 0.040 @7.4V	60	https://www.freakware.de/p/hitec-hs-8315bh-hit-114315-a85753.htm
Yantrs 0427AS-MGX-E	6.0-8.4	6.0	20 @6.0V 23 @7.4V 27 @8.4V	0.058 @6.0V 0.047 @7.4V 0.040 @8.4V	70	https://de.aliexpress.com/item/1005005224149218.html

Servo Type	Voltage [V]	Current [A]	Torque [kg*cm]	Speed [s/60°]	Weight [g]	Notes and Links
GX_Servo X15BLS	6.0-8.4		12 @6.0V 15 @7.4V 18 @8.4V	0.070 @6.0V 0.060 @7.4V 0.050 @8.4V	70	https://de.aliexpress.com/item/1005004649121319.html
GX_Servo X20BLS			18 @6.0V 20 @7.4V 22 @8.4V	0.060 @6.0V 0.050 @7.4V 0.045 @8.4V		
GX_Servo X25BLS			22 @6.0V 25 @7.4V 28 @8.4V	0.060 @6.0V 0.050 @7.4V 0.045 @8.4V		
Savöx SH-1290MG	4.8-6.0		4.0 @4.8V 5.0 @6.0V	0.059 @4.8V 0.048 @6.0V	57	https://www.savox-shop.com/de/savoex-sh-1290mg-servo.html
Real Hawk?	3.7-4.2	0.035	0.065 @3.7V 0.075 @4.2V	0.070 @3.7V 0.050 @4.2V	1.9	ultra small https://de.aliexpress.com/item/33006571747.html
A06CLS	4.8-8.4		1.8 @4.8V 2.2 @6.0V 2.6 @7.4V 3.0 @8.4V	0.075 @4.8V 0.066 @6.0V 0.058 @7.4V 0.052 @8.4V	7	ultra small https://de.aliexpress.com/item/1005005104844054.html
DS820M	7.4-8.4		22 @7.4V 23 @8.4V	0.060 @7.4V 0.055 @8.4V	80	https://www.freakware.de/p/ds820m-high-voltage-brushless-servo-align-hsd82001-a147728.htm
Yantrs 0640AS-MGX-E	6.0-8.4	6.0	32 @6.0V 37 @7.4V 40 @8.4V	0.080 @6.0V 0.070 @7.4V 0.060 @8.4V	70	https://de.aliexpress.com/item/1005005423898270.html
Yantrs 0612MG-B	6.0-8.4	2.8	9 @6.0V 10.5 @7.4V 12 @8.4V	0.090 @6.0V 0.070 @7.4V 0.060 @8.4V	22	https://de.aliexpress.com/item/1005005381188231.html
Sirenxi?	3.7-6.0		0.2 @4.8V	0.080 @4.8V	3.1	ultra small https://de.aliexpress.com/item/1005005652463467.html
DS3240	4.8-6.8		36 @5.0V 45 @6.8V	0.200 @5.0V 0.170 @6.8V	68.3	https://de.aliexpress.com/item/1005005857192248.html
SG90	4.2-6.0		1.6 @4.8V	0.300 or 0.120 ?	10.1	very cheap https://de.aliexpress.com/item/1005005857192248.html

Servo Type	Voltage [V]	Current [A]	Torque [kg*cm]	Speed [s/60°]	Weight [g]	Notes and Links
ASMC-03B	11-30	?	180	0.50 @24V	530	ultra high torque https://de.aliexpress.com/item/4000322402143.html
ASMC-04A	11-30	3.0	110	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005006010596284.html
ASMC-04B			180	0.50 @24V		
ASMC-05A	11-30	3.0	110 @24V	0.12 @24V	530	ultra high torque https://de.aliexpress.com/item/32889090668.html https://de.aliexpress.com/item/32820735038.html
ASMC-05B			180 @24V	0.50 @24V		
ASMC-LQB	8-12	1.0 ?	60	0.50 ?	170	https://de.aliexpress.com/item/1005004137170731.html https://de.aliexpress.com/item/1005005412800200.html
ASME-02						https://content.instructables.com/FCA/QIQI/IQA4XCJY/FCAQIQIIQA4XCJY.pdf
ASME-04A	11-30	5.0	220 @12V	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005006010596284.html
ASME-04B			340 @24V	0.50 @24V		
ASME-05B	11-30	5.0	380 @24V	0.50 @24V	560	ultra high torque https://de.aliexpress.com/item/32973185578.html
ASME-MRA	11-30	5.0	110 or 260?	0.12 @24V	560	ultra high torque https://de.aliexpress.com/item/1005005633621592.html https://de.aliexpress.com/item/1005005417403122.html https://de.aliexpress.com/item/1005004126382458.html
ASME-MRB			380 or 400?	0.50 @24V		
ASME-MXA	11-30	5.0	130 @12V 260 @24V	0.24 @12V 0.12 @24V	550	ultra high torque with 10 revolutions https://de.aliexpress.com/item/1005006010596284.html https://de.aliexpress.com/item/1005006236191122.html
ASME-MXB			190 @12V 380 @24V	1.00 @12V 0.50 @24V		
ASME-SDA	7-30	5.0	90	0.12 @24V	450	ultra high torque https://de.aliexpress.com/item/1005004756047692.html
ASME-SDB			180	0.50 @24V		
ASME-SQA	7-24	1.0 ?	90 @24V	0.12 @24V	400	ultra high torque https://de.aliexpress.com/item/1005004137302220.html
ASME-SQB			180 @24V	0.50 @24V		
ASMG-MTA	11-30	10.0	260 @24V	0.12 @24V	550	ultra high torque

Servo Type	Voltage [V]	Current [A]	Torque [kg*cm]	Speed [s/60°]	Weight [g]	Notes and Links
ASMG-MTB			500 @24V	0.50 @24V		https://de.aliexpress.com/item/1005005440128075.html https://de.aliexpress.com/item/1005006310483520.html
DHLG-03X	24-30	20	700 or 1000 ?	0.50 @24V	4500 or 6300 ?	ultra high torque https://de.aliexpress.com/item/1005004305694514.html https://de.aliexpress.com/item/1005005212645904.html

Hacking the firmware for Wingxine (ASME) servos: https://hackaday.io/project/19082-wingxine-asme-03-alternative-firmware

Servo Controllers:

Type	Voltage [V]	Current [A]	Notes and Links
ASMF-04	7-30	10	with feedback potentiometer https://de.aliexpress.com/item/1005004137043781.html
ASMB-03	8-48	20	with feedback potentiometer https://de.aliexpress.com/item/1005004133955023.html https://www.amazon.de/Okuyonic-Servomotor-Controller-Servomotor-Treiberplatine-Servocontroller-Platine-Servomontage/dp/B09YP7TF94 https://www.alibaba.com/product-detail/DIY-8V-48V-20A-1000N-m_60492358055.html
ASMB-03M	8-48	20	with magnetic feedback sensor https://de.aliexpress.com/item/1005004133955023.html
ASMB-04	8-48	20	with feedback potentiometer https://de.aliexpress.com/item/1005006519077313.html https://www.ebay.de/itm/155281102015
ASMB-03M	8-48	20	with magnetic feedback sensor https://de.aliexpress.com/item/1005004133955023.html

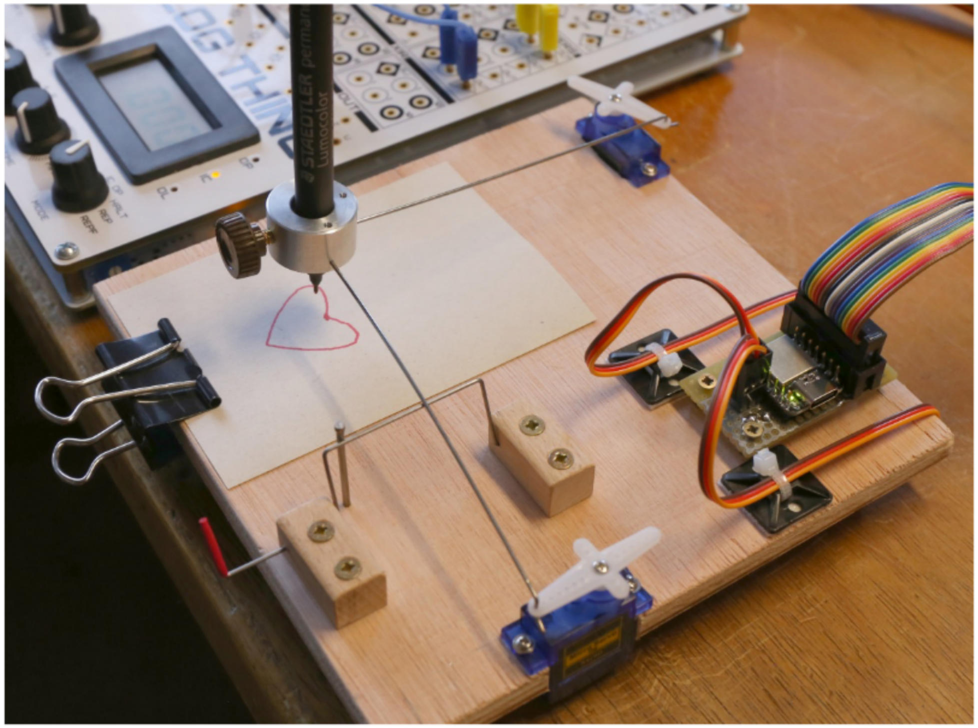
What maximum (sine) frequency can be expected at the servo axis?

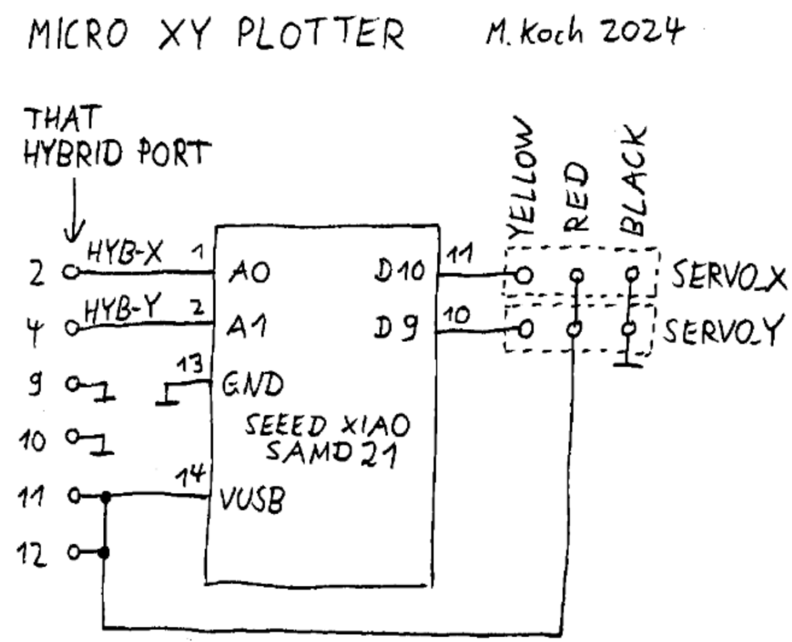
$F_{max} = 1 / (2 * pi * T)$

Time for 60° Movement	F_{max} for 60° Peak Amplitude (120° Peak to Valley)	F_{max} for 30° Peak Amplitude (60° Peak to Valley)
0.02 s	7.96 Hz	15.92 Hz
0.025 s	6.37 Hz	12.73 Hz
0.03 s	5.31 Hz	10.61 Hz
0.035 s	4.55 Hz	9.09 Hz
0.04 s	3.98 Hz	7.96 Hz
0.05 s	3.18 Hz	6.37 Hz
0.06 s	2.65 Hz	5.31 Hz
0.08 s	1.99 Hz	3.98 Hz
0.10 s	1.59 Hz	3.18 Hz
0.20 s	0.80 Hz	1.59 Hz
0.30 s	0.53 Hz	1.06 Hz

4.9 Micro XY Plotter for Hybrid Port

This is a very simple XY plotter for the hybrid port, realized with a Seeed XIAO SAMD21 microcontroller and two SG90 servos. It seems that level conversion from 3.3V to 5.0V is not required for these servos. Of couse, the X and Y position of the pen is a nonlinear function of the servo positions. The linearization is done in software, so that the pen position is proportional to the X and Y signals in THAT..





Software for Seeed XIAO SAMD21 microcontroller in XY plotter:

```
//  THAT XY Plotter
//  Michael Koch, January 2024

#include <Servo.h>

Servo myservo1, myservo2;
float x, y, b, c, d, e, xx, yy, alpha, beta;
const float a = 17.2;          // length of servo arm in mm, from servo axis to hinge
const float r = 114;           // length of rod in mm, from pen to hinge
const float p = 13;            // half of drawing range in mm (scale factor mm/MU)
const float gain = 0.000809;   // gain for ADU to MU conversion
const float offset = -1.656;   // offset for ADU to MU conversion

void setup()  // the setup function runs once when you press reset or power the board
{
  analogReadResolution(12);
  myservo1.attach(D10);
  myservo2.attach(D9);
}

void loop()   // the loop function runs over and over again forever
{
  x = offset + gain * analogRead(A0);   // -1 to +1
  y = offset + gain * analogRead(A1);   // -1 to +1
  x = x * p;                            // convert to mm
  y = y * p;                            // convert to mm
  b = (y - a) / (r - x);
  c = (x * x + y * y - 2 * r * r) / (2 * (x - r));
  d = (2 * b * (c - x) - 2 * y) / (1 + b * b);
  e = (x * x - 2 * x * c + c * c + y * y - r * r) / (1 + b * b);
  yy = -d / 2 - sqrt(d * d / 4 - e);    // coordinates of hinge
  xx = yy * b + c;                      //  of X axis servo
  alpha = atan((xx - r) / (a - yy));    // X servo angle
  b = (x + a) / (y - r);
  c = (x * x + y * y - 2 * r * r) / (2 * (y - r));
  d = (2 * b * (c - y) + 2 * x) / (1 + b * b);
  e = (y * y - 2 * y * c + c * c + x * x - r * r) / (1 + b * b);
  yy = -d / 2 - sqrt(d * d / 4 - e);    // coordinates of hinge
  xx = yy * b + c;                      //  of Y axis servo
  beta = atan((xx - r) / (a - yy));     // Y servo angle
  myservo1.writeMicroseconds(1500 + (int)(alpha / 3.1415 * 2000));  // convert from radians to pulse width
  myservo2.writeMicroseconds(1500 + (int)(beta / 3.1415 * 2000));   // convert from radians to pulse width
}
```

How does the linearization work?

The origin of the coordinate system is in the center of the plotting area.

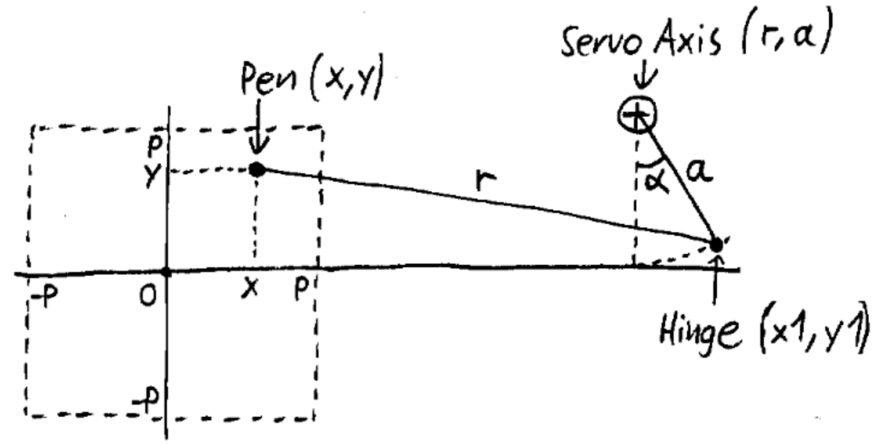
(x,y) is the known pen position.

(r,a) is the known position of the servo axis for the x-axis movement.

a is the length of the servo arm (from axis to hinge).

r is the length from the hinge to the pen.

(x1,y1) is the unknown position of the hinge.



We have two given equations:

$(x - x1)^2 + (y - y1)^2 = r^2$ and $(r - x1)^2 + (a - y1)^2 = a^2$

After subtracting the second equation from the first one and solving for x1, we get:

$x1 = y1 * b + c$ with $b = (y - a) / (r - x)$ and $c = (x^2 + y^2 - 2 * r^2) / (2 * (x - r))$

Insert this x1 into the first equation to get the solution for y1:

$y1 = -d / 2 - \mathrm{sqrt}(d * d / 4 - e)$ with $d = (2 * (b * c - x * b - y)) / (1 + b^2)$ and $e = (x^2 - 2 * x * c + c^2 + y^2 - r^2) / (1 + b^2)$

and then we get

$x1 = y1 * b + c$

The angle of the servo axis is $alpha = \mathrm{atan}((x1 - r) / (a - y1))$

For the y-axis servo you can use the same equations after replacing x by y and y by -x.

Pulse width for SG90 servo: 500µs -90°, 1000µs -45°, 1500µs 0°, 2000µs +45°, 2500µs +90°

4.10 Ideas for Hybrid Computer with Teensy 4.0

Wishes:

- Keep it simple
- Digital potentiometers are typically set before the computer is started, and not changed while the computer is running. Most users will be happy with statically set potentiometers (but I'm not)
- Stand-alone usage is important for many users. Especially for schools. Best if it could be used without computer and without oscilloscope.
  Problem: Stand-alone means a formula parser must be implemented. That's not easy, and it's slower than having the user-defined functions as compiled C code.
  The alternative is to use the Arduino/Teensy IDE for implementing the user-defined functions. The source code can be split in two files: One file for user-defined things, and the other file for the operating system.
- There are good color OLED displays with touch interface
- Additionally the hybrid part can be controlled over USB interface from a PC, using a simple command language. For example for setting of the digital potentiometers.
- AD5293 has 10-bit values, set via SPI. Conversion from [0..1] to [0..1023]
- +-1 Machine unit = +-10V --> 12-bit 4.9mV, 14-bit 1.2mV, 16-bit 0.3mV --> 14-bit is sufficient!

Ideas how the digital potentiometers (or DACs) can be controlled:

- Statically set a value from the PC via USB interface, using a simple command language.
- Function generator with one input: out=f(x) where x is a dynamic input signal from a ADC, realised with look-up-table, updated with constant sample rate.
- Function generator with multiple inputs: out=f(x,y,...) where x,y,... are dynamic input signals from ADCs, calculation in realtime, a constant sample rate can't be guaranteed.
- Signal generator with constant frequency: out=f(omega * t) with selectable frequency and waveform (sine, square, triangle, sawtooth, ...) and selectable phase shift
- Signal generator with variable frequency: out=f(x * omega * t) where x is a dynamic input signal from a ADC, with selectable frequency and waveform (sine, square, triangle, sawtooth, ...) and selectable phase shift. No phase jump when the frequency is changed.

Comparison of microcontroller boards:

	ADCs	DACs	RAM	Flash ROM	CPU Clock	Microcontroller	Notes and Links
Teensy 2.0	12 x 10-bit	-	2.5 kB	31.5 kB	16 MHz	ATMEGA32U4 AVR	no longer available https://www.pjrc.com/store/teensy.html
Teensy LC	13 x 16-bit	1 x 12-bit	8 kB	62 kB	48 MHz	MKL26Z64VFT4 Cortex-M0+	no longer available https://www.pjrc.com/store/teensylc.html
Teensy 4.0	14 x 12-bit	-	1 MB	1984 kB	600 MHz	IMXRT1062DVL6 Cortex-M7	Floating point math unit, 64 & 32 bits https://www.pjrc.com/store/teensy40.html
Teensy 4.1	18 x 12-bit	-	1 MB + 2*8 MB	7936 kB	600 MHz	IMXRT1062DVJ6 Cortex-M7	Floating point math unit, 64 & 32 bits https://www.pjrc.com/store/teensy41.html
Arduino Nano ESP32	8 x	-	512 kB	384 kB	up to 240MHz	ESP32-S3	https://store.arduino.cc/collections/boards-modules/products/nano-esp32
Arduino Nano 33 IoT	8 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	SAMD21 Cortex®-M0+	https://store.arduino.cc/products/arduino-nano-33-iot
Arduino UNO R4 Minima	6 x 14-bit	1 x 12-bit	32 kB	256 kB	48 MHz	Renesas RA4M1	Has a single-precision FPU https://store.arduino.cc/collections/boards-modules/products/uno-r4-minima
Arduino Due	12 x 12-bit	2 x 12-bit	96 kB	512 kB	84 MHz	Atmel SAM3X8E ARM Cortex-M3 CPU	https://store.arduino.cc/collections/boards-modules/products/arduino-due-without-headers DAC example: https://docs.arduino.cc/tutorials/due/simple-waveform-generator/
Arduino Giga R1 Wifi	12 inputs, 3 16-bit ADCs	2 x 12-bit	1 MB	2 MB	480 MHz and 240 MHz	Dual core Cortex®-M7 Cortex®-M4	https://store.arduino.cc/products/giga-r1-wifi 800x480 Display: https://docs.arduino.cc/resources/datasheets/ASX00039-datasheet.pdf
Portenta H7 Lite	11 inputs, 3 16-bit ADCs	2 x 12-bit but only one has a pin?	1 MB	2 MB	480 MHz and 240 MHz	STM32H747 dual core Cortex® M7 Cortex® M4	https://store.arduino.cc/products/portenta-h7-lite https://docs.arduino.cc/resources/datasheets/ABX00042-ABX00045-ABX00046-datasheet.pdf
Seeed XIAO SAMD21	11 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	ATSAMD21G18A-MU	https://wiki.seeedstudio.com/Seeeduino-XIAO/
Arduino Zero SAMD21	6 x 12-bit	1 x 10-bit	32 kb	256 kB	48 MHz	ATSAMD21G18, 32-Bit ARM® Cortex® M0+	https://store.arduino.cc/products/arduino-zero

GY SAMD21	14 x 12-bit	yes	32 kB	256 kB	48 MHz	ATSAMD21G18	https://de.aliexpress.com/i/32975308152.html I found no documentation at all!
Sparkfun SAMD21	14 x 12-bit	1 x 10-bit	32 kB	256 kB	48 MHz	ATSAMD21G18	https://www.sparkfun.com/products/13664 Some documentation: https://learn.sparkfun.com/tutorials/samd21-minidev-breakout-hookup-guide
JOY-IT NodeMCU ESP32	15 x 12-bit	2 x 8-bit	512 kB	4 MB	240 MHz	Tensilica LX6 Dual-Core	https://joy-it.net/de/products/SBC-NodeMCU-ESP32 I didn't find much documentation for this board! DAC resolution is 8-bit as described here: https://circuits4you.com/2018/12/31/esp32-dac-example/ ADC resolution is 12-bit: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
ESP-WROOM-32	18 x 12-bit	2 x 8-bit					https://de.aliexpress.com/item/1005006474252674.html https://www.elektor.de/amfile/file/download/file/1674/product/8641/

Teensy tech specs: https://www.pjrc.com/teensy/techspecs.html

8-Channel ADC chips with SPI output:

Chip	Channels	Resolution	SR	Package	Price and Link	Datasheet	Notes
AD7606BSTZ	8	16-bit	200kSPS	LQFP 64 0.5mm	38.43 EUR https://de.farnell.com/analog-devices/ad7606bstz/a-d-wandler-16-bit-200ksps-lqfp/dp/4016556?st=ad7606 Evaluation Board: https://de.farnell.com/analog-devices/eval-ad7606bfmcz/evaluationsboard-adc/dp/4032769?st=ad7606 Very cheap board from Aliexpress: https://de.aliexpress.com/item/4001307685007.html https://de.aliexpress.com/item/1005006303666676.html	https://www.analog.com/media/en/technical-documentation/data-sheets/ad7606_7606-6_7606-4.pdf Evaluation Board: https://www.analog.com/media/en/technical-documentation/user-guides/eval-ad7606bfmcz-ug-1225.pdf Problem: There is no SPI library for Teensy with more than 1 data line. Either use SPI with one data line (reduced throughput) or 16-bit parallel interface.	simultaneous Sampling, +-10V Bipolar Input
AD7607	8	14-bit	200kSPS	LQFP 64 0.5mm	32.68 EUR https://de.farnell.com/analog-devices/ad7607bstz/a-d-wandler-14-bit-200ksps-lqfp/dp/4016565?ost=ad7607	https://www.analog.com/media/en/technical-documentation/data-sheets/ad7607.pdf	Simultaneous Sampling, +-10V Bipolar Input
AD7761	8	16-bit	256kSPS	LQFP 64 0.5mm ???	20.48 EUR https://de.farnell.com/analog-devices/ad7761bstz/a-d-wandler-16bit-256ksps-lqfp/dp/4016745?ost=ad7761	https://www.analog.com/media/en/technical-documentation/data-sheets/AD7761.pdf	Simultaneous Sampling, Sigma-Delta, complicated
ADAS3023	8	16-bit	125kSPS	LFCSP40 0.5mm	36.88 EUR https://de.farnell.com/analog-devices/adas3023bcpz/a-d-wandler-16bit-125ksps-lfcsp/dp/4017219?ost=adas3023	https://www.analog.com/en/products/adas3023.html	Simultaneous Sampling
LTC2345-16	8	16-bit	200kSPS	?	?	https://www.analog.com/media/en/technical-documentation/data-sheets/234516f.pdf	Differential inputs
LTC2358-	8	16-bit	200kSPS	LQFP 48	39.24 EUR	https://www.analog.com/media/en/t	Differential

16				**0.5mm**	**https://de.farnell.com/analog-devices/ltc2358clx-16-pbf/a-d-wandler-16-bit-200ksps-lqfp/dp/4018751?st=ltc2358**	**echnical-documentation/data-sheets/ltc2358-16.pdf**	**±10.24V**

Sample code for Teensy for reading AD7606 with SPI interface (one data line):

Source: **https://forum.pjrc.com/index.php?threads/spi-with-2-or-4-data-lines.74015/**

```
#include <SPI.h>

uint32_t startTime, elapsedTime, old_res;
#define SCALE_FACTOR 0.000152587890625
/* D14 and D15 should be grounded if you use this ad7606 board
 *  https://www.amazon.de/RELAND-SUN-Multi-Channel-Datenerfassungsmodul-Synchronisation/dp/B0B4HKSK8B/ref=sr_1_2?keywords=ad7606&qid=1681803018&sr=8-2
 *  i do not use oversampling in this example
 *  i set range to the ground in this example so the range is +/-5V
*/
#define BUSY 32               // you can use any digital pin
#define RESET 30              // you can use any digital pin
#define START_CONVERSION 34   // you can use any digital pin
#define CHIP_SELECT 10        // SPI CS     // #define CHIP_SELECT 53        // SPI CS
#define D7_out 12             // SPI MISO there is no need to use MOSI port with the ad7606 #define D7_out 50             // SPI MISO there is no need to use MOSI port with the ad7606
#define RD 13                 // SPI SCLK #define RD 52                 // SPI SCLK
#define RANGE 36               // you can use any digital pin
#define TOTAL_RAW_BYTES 16
SPISettings _spiSettings;

int bytesToRead = TOTAL_RAW_BYTES;
byte raw[TOTAL_RAW_BYTES];
uint16_t parsed[8];


//--------------SETUP-UP--------------------------
void setup()
{
    initial();
    Serial.begin(2000000);
    while (!Serial) {} // wait for usb connection
    SPI.begin();
}
//---------------LOOP-------------------------
void loop()
{
    startTime = micros();
    readData();
    elapsedTime = micros() - startTime;
    if (elapsedTime != old_res)
    {
```

```
        Serial.print("Conversion time:\t");
        Serial.print(elapsedTime);
        Serial.println(" microseconds");
        old_res = elapsedTime;
    }
    parseRawBytes();

    for (int i = 0; i < 8; i++) {
        Serial.print((float)parsed[i] * SCALE_FACTOR, 5);
        Serial.print(",");
    }

    Serial.print("\r\n");
}
//-------------------------------------------
void initial()
{
    pinMode(BUSY, INPUT);
    pinMode(RESET, OUTPUT);
    pinMode(START_CONVERSION, OUTPUT);
    pinMode(CHIP_SELECT, OUTPUT);
    pinMode(D7_out, OUTPUT);
    pinMode(RD, OUTPUT);
    digitalWrite(START_CONVERSION, HIGH);
    digitalWrite(CHIP_SELECT, HIGH);
    reset(RESET);
}
//-------------------------------------------
void parseRawBytes() {
    /*
    I figured the problem should be here in the function of parseRawBytes.
    The sizeof(int) is 4 in Portanta and also Giga.
    This results in the value of (sizeof(parsed) / sizeof(int)) is 4 not 8,
    so only the data for the first 4 channels can be obtained.
    I modify the maximum value of loop, eight channels data can be obtained.

        for (int i = 0; i < (sizeof(parsed) / sizeof(int)); i++)
        {
            parsed[i] = (raw[i * 2] << 8) + raw[(i * 2) + 1];
        }
    */

    for (int i = 0; i < 8; i++)
    {
        parsed[i] = (raw[i * 2] << 8) + raw[(i * 2) + 1];
    }
}
//-------------------------------------------
/*reset signal*/
void reset(uint8_t port)
{
    digitalWrite(port, HIGH);
```

```
    // delayMicroseconds(1);
    digitalWrite(port, LOW);
    //  delayMicroseconds(1);
}
//-------------------------------------------
void conversionPulse(uint8_t port)
{
    digitalWrite(port, LOW);
    //  delayMicroseconds(1);
    digitalWrite(port, HIGH);
}
//-------------------------------------------
/*
1- start conversion START_CONVERSION HIGH
2- wait until the busy is LOW again
3- put the CS to LOW
4- Read a byte from the SPI
*/
void readData()
{
    conversionPulse(START_CONVERSION);
    while (digitalRead(BUSY) == HIGH) {
        //  delayMicroseconds(1);
    }
    SPI.beginTransaction(_spiSettings);
    digitalWrite(CHIP_SELECT, LOW);
    while (bytesToRead > 0) {
        raw[TOTAL_RAW_BYTES - bytesToRead] = SPI.transfer(0x00);
        bytesToRead--;
    }
    digitalWrite(CHIP_SELECT, HIGH);
    SPI.endTransaction();

    bytesToRead = TOTAL_RAW_BYTES;
}
//-------------------------------------------
/*
  0 +/-  5V
  1 +/- 10V
*/
void setRange(bool range)
{
    pinMode(RANGE, OUTPUT);
    digitalWrite(RANGE, range);
}
//-------------------------------------------
/*OS2 OS1 OS0
oversampling
000 No oversampling Maximum sampling rate is 200KSPS
001  2 times        100 KSPS
010  4 times        50 KSPS
011  8 times        25 KSPS
```

```
100 16 times        12.5 KSPS
101 32 times        6.25 KSPS
110 64 times        3.125 KSPS
*/
/*
void setOversampling(uint8_t OS0, uint8_t OS1, uint8_t OS2) {
    pinMode(_OS0, OUTPUT);
    pinMode(_OS1, OUTPUT);
    pinMode(_OS2, OUTPUT);
    digitalWrite(_OS0, bitRead(B001, OS0));
    digitalWrite(_OS1, bitRead(B010, OS1));
    digitalWrite(_OS2, bitRead(B100, OS2));
}
*/
```

GPIO6_PSR is the hardware register that must be read:

uint32_t port6_val = GPIO6_PSR;

```
//MXRT pin  |6.31|6.30|6.29|6.28|6.27|6.26|6.25|6.24|6.23|6.22|6.21|6.20|6.19|6.18|6.17|6.16|
//4.1 pins  |27  |26  |39  |38  |21  |20  |23  |22  |16  |17  |41  |40  |15  |14  |18  |19  |
```

If you can just leave the 16 pins as input mode, I would recommend first configuring all 16 with ordinary pinMode(). There are important low-level details about pin configuration which are handled automatically by pinMode().

sMore interesting infos from Pauf Stoffregen: (Source: https://forum.pjrc.com/index.php?threads/spi-with-2-or-4-data-lines.74015/)

In the reference manual you'll find a lot of documentation which says these pins are connected to GPIO1. Then elsewhere you can find info about the ability to map a pin to either the slow GPIO (1 to 5) or fast GPIO (6 to 9). But almost all the documentation was written for the earlier RT1052 chip, so a lot of documentation mentions GPIO1 only. Of course, knowing how the RT1062 hardware works, you can know that any pins GPIO1 accesses can alternately be accessed by GPIO6.

Teensy's startup code maps all the pins to the fast GPIO peripherals. So even though the default for non-Teensy boards would be GPIO1, because of the way the Teensy core library is designed, you would access the port with GPIO6 rather than GPIO1.

Not documented in the reference manual is the (likely) story behind *why* the chip has redundant GPIO peripherals. NXP's earlier RT1052 chip had only the slow GPIO. They almost certainly heard feedback from (huge corporation) customers about the slow GPIO performance. In the design of these modern chips, it's much easier to bolt on another peripheral than to dive deep down into the design of the existing ones. And the slowness comes from various buses and bus bridges, which might be IP they licensed from ARM which often comes with restrictions on modifications (which is a smart policy for ARM as it keeps a lot of consistency in the platform and required compiler support). So rather than do anything about the GPIO slowness, they just added another copy of the same GPIO perhipheral but connected it to a special low-latency bus the M7 core provides. And then they added muxes to let you assign each pin to either the slow or fast GPIO.

The main downside to the fast GPIO is DMA can't access anything on the low latency bus (all 0x42xxxxxx addresses). So when a pin needs to accessed by DMA, as the OctoWS2811 library does, the special mux registers are used to assign those pins to the slow GPIO which DMA can access.

Other than DMA, you would normally always want to the fast GPIO. That's why Teensy's startup code assigns all the pins to be used by fast GPIO6 to GPIO9.

DAC chips with SPI input:

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5363	8	14-Bit	+-10V		???	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5362_5363.pdf	
AD5544	4	16-bit	Current output, needs external OP-Amp	SSOP 28	53.49 EUR https://de.farnell.com/analog-devices/ad5544arsz-reel7/d-a-wandler-16bit-ssop-28-40-bis/dp/4023048?st=ad5544	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5544_5554.pdf	Multiplying DAC, Vref=+-10V Compatible with DAC8814
DAC8814	4	16-bit	Current output, needs external OP-Amp	SSOP 28	34.40 EUR https://www.digikey.de/de/products/detail/texas-instruments/DAC8814ICDBT/863244	https://www.ti.com/lit/ds/symlink/dac8814.pdf	Multiplying DAC, Vref=+-10V Compatible with AD5544
AD5554	4	14-Bit	Current output, needs external OP-Amp	SSOP 28	38.89 EUR https://de.farnell.com/analog-devices/ad5554brsz/d-a-wandler-14bit-ssop-28-40-bis/dp/4023062?ost=ad5554	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5544_5554.pdf	Multiplying DAC Vref=+-10V Compatible with DAC8803
DAC8803	4	14-Bit	Current output, needs external OP-Amp	SSOP 28	25.83 EUR https://www.mouser.de/ProductDetail/Texas-Instruments/DAC8803IDBT?qs=vul0MIC%2Fa1cIPELYpZUEyA%3D%3D	https://www.ti.com/lit/ds/symlink/dac8803.pdf	Multiplying DAC Vref=+-10V Compatible with AD5554
AD5648	8	14-Bit	0 - 2.5V or 0 - 5.0V	TSSOP 16 0.65mm	22.41 EUR https://de.farnell.com/analog-devices/ad5648aruz-2/d-a-wandler-14bit-tssop-16-40/dp/4017484?st=ad5648	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5628_5648_5668.pdf	with 5 ppm/°C, On-Chip Reference

AD5668	8	16-Bit	0 - 2.5V or 0 - 5.0V	TSSOP 16 0.65mm	26.18 EUR https://de.farnell.com/analog-devices/ad5668aruz-2/d-a-wandler-16bit-tssop-16-40/dp/4017526?st=ad5668	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5628_5648_5668.pdf Arduino Library: https://www.arduino.cc/reference/en/libraries/ad56x8/	with 5 ppm/°C, On-Chip Reference
AD5676	8	16-Bit		TSSOP 20 0.65mm	22.91 EUR https://de.farnell.com/analog-devices/ad5676bruz/d-a-wandler-16bit-tssop/dp/4017544?st=ad5676	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5676.pdf	no on-chip reference
AD5678	4+4	4×12-Bit + 4×16-Bit		TSSOP 14 0.65mm	???	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5678.pdf	On-Chip Reference
LTC2600	8	16-Bit	Rail-to-Rail	SSOP 16 0.635mm	28.18 EUR https://de.farnell.com/analog-devices/ltc2600cgn-trpbf/d-a-wandler-16bit-nssop-16/dp/4218020?st=ltc2600	https://www.analog.com/media/en/technical-documentation/data-sheets/2600fe.pdf	
LTC2656	8	16-bit or 12-bit	Rail-to-Rail		I found only the 12-bit version	https://www.analog.com/media/en/technical-documentation/data-sheets/2656fa.pdf	with 10ppm/°C Max Ref

A/D and D/A combinations:

Chip	Channels	Resolution	Output	Package	Price and Link	Datasheet	Notes
AD5592R	8	12-bit	5V	TSSOP16 0.65mm	9.12 EUR https://www.mouser.de/ProductDetail/Analog-Devices/AD5592RBRUZ?qs=wFnXgzJ2EGM0E%2FJNKZBaug%3D%3D 9.16 EUR https://de.farnell.com/analog-devices/ad5592rbruz/a-d-wandler-12-bit-400ksps-tssop/dp/4016360?st=ad5592r	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5592r.pdf Library for Teensy: https://github.com/dzalf/AD5592R	SPI interface, contains a total of 8 ADCs or DACs
AD5593R	8	12-Bit	5V	TSSOP16 0.65mm	https://de.aliexpress.com/item/1005002433701414.html 9.19 EUR https://de.farnell.com/analog-devices/ad5593rbruz-rl7/konfigurierbarer-adc-dac-12bit/dp/4016365?st=ad5593r	https://www.analog.com/media/en/technical-documentation/data-sheets/ad5593r.pdf	I2C interface, contains a total of 8 ADCs or DACs

Digital potentiometers:

Chip	Positions	R	Package	Price and Link	Datasheet
AD5293	1024 (10-bit)	20k, 50k or 100k	TSSOP 14	8.12 EUR 10 pieces: 7.32 EUR https://de.farnell.com/analog-devices/ad5293bruz-50/digitalpotentiometer-1fach-tssop/dp/4028695?st=ad5293	https://www.analog.com/media/en/technical-documentation/data-sheets/AD5293.pdf

Displays:

Color 320x240 TFT Touchscreen, ILI9341 Controller Chip: https://www.pjrc.com/store/display_ili9341_touch.html

16-bit I2C I/O Expander: MCP23017

Arduino / Teensy SPI Library: **https://www.arduino.cc/reference/en/language/functions/communication/spi/**

Teensy 4.0 and 4.1 has 1MB RAM. It can be extended by 8MB or 16MB with this chip: **https://www.pjrc.com/store/psram.html**

Which pins from THATs hybrid port should be used?

Pin	Name	Function	Use it?
2	HYB-X	low impedance analog output signals, voltage is 0.1 * OUT-X + 1.64V, range 0.64V to 2.64V for +-1 MU	These output signals can be used for an oscilloscope.
4	HYB-Y		
6	HYB-Z		
8	HYB-U		
1	IN-X	The purpose of these signals is unclear.	no
3	IN-Y		
5	IN-Z		
7	IN-U		
9, 10	GND	ground	yes
11, 12	VUSB	is connected to +5V USB supply voltage via 750mA PTC fuse	no, the hybrid part should have its own USB supply.
13	DIR	This is an input with 3.3V level, low level enables THATs hybrid mode. The input has an internal pullup resistor to 3.3V and it not 5V tolerant.	yes
14	-Mode_OP	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = operate In hybrid mode: 5V tolerant 3.3V input, sets THATs operation mode, the input has no internal pullup resistor	yes
15	Voffset	+1.64V	no
16	-Mode_IC	In normal mode: Output (via 5k1 resistor) with 3.3V level, low = initial condition In hybrid mode: 5V tolerant 3.3V input, sets THATs initial condition mode, the input has no internal pullup resistor	yes

Note: Looking from the rear side into the hybrid port: Pin 1 is top right, Pin 2 is bottom right.

How the hybrid port works:

DIR must be pulled low to set THAT in "Hybrid" mode. The position of the MODE switch doesn't care.

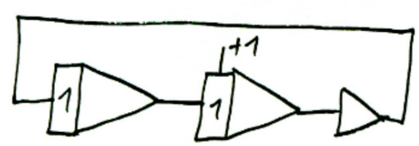
Low level at -ModeOP sets THAT in "Operate" mode.

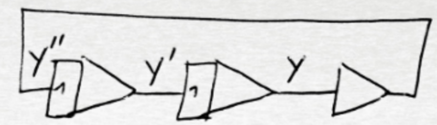
Low level at -ModeIC sets THAT in "Initial Condition" mode.

5 Signal Generators

5.1 Sine Generator

How can the frequency of this sine generator be calculated?





$$y = \sin(2\cdot\pi\cdot f\cdot t)$$

$$y' = \cos(2\cdot\pi\cdot f\cdot t)\cdot 2\cdot\pi\cdot f \cdot 1ms$$

$$y'' = -\sin(2\cdot\pi\cdot f\cdot t)\cdot(2\cdot\pi\cdot f)^2\cdot(1ms)^2$$

$$y = -y'' \Rightarrow (2\pi f)^2\cdot(1ms)^2 = 1$$

$$\Rightarrow 2\pi f \cdot 1ms = 1$$

$$\Rightarrow 2\pi f = 1kHz$$

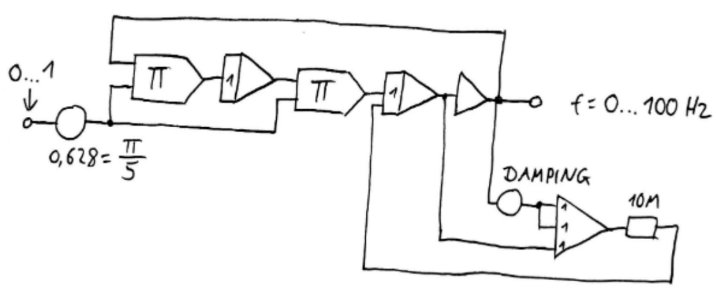
$$\Rightarrow f = \frac{1kHz}{2\pi}$$

5.2 Voltage controlled Sine Generator

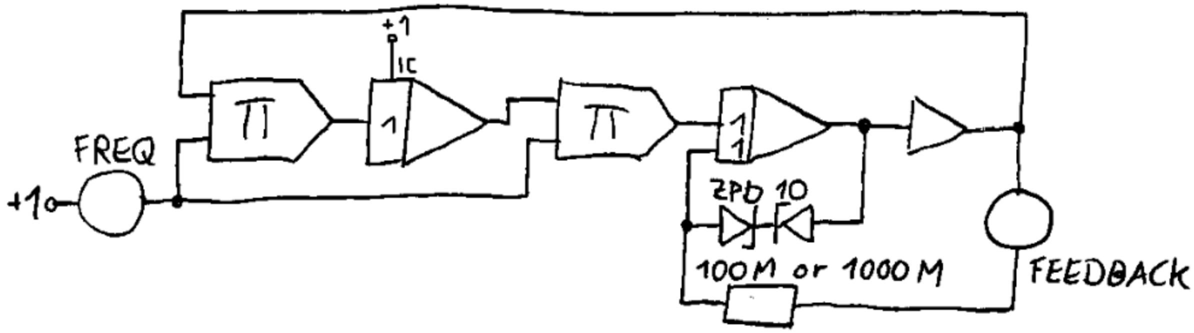
Forgot +1 initial condition at one of the integrators. The amplitude will slowly run away towards zero or overflow. This can be adjusted with the DAMPING coefficient. This is not amplitude stabilization. It does only allow for a longer time before the amplitude drifts away.

Integration Resistor	Integration Capacitor	Input = 0.628 (0.2*Π)		Input = 1	
		Frequency	Period	Frequency	Period
100 kΩ ("10" Input)	1nF (normal)	1 kHz	1 ms	1592 Hz (5000/Π) Hz	628 µs (200*Π) µs
1 MΩ ("1" Input)	1nF (normal)	100 Hz	10 ms	159 Hz (500/Π) Hz	6.28 ms (2*Π) ms
1 MΩ ("1" Input)	10nF (external)	10 Hz	100 ms	15.9 Hz (50/Π) Hz	62.8 ms (20*Π) ms
1 MΩ ("1" Input)	100nF (SLOW)	1 Hz	1 s	1.59 Hz (5/Π) Hz	628 ms (200*Π) ms
10 MΩ (external)	100nF (SLOW)	100 mHz	10 s	159 mHz (500/Π) mHz	6.28 s (2*Π) s
100 MΩ (external)	100nF (SLOW)	10 mHz	100 s	15.9 mHz (50/Π) mHz	62.8 s (20*Π) s
1 GΩ (external)	100nF (SLOW)	1 mHz	1000 s	1.59 mHz (5/Π) mHz	628 s (200*Π) s
1 GΩ (external)	1µF (with open amplifier)	100 µHz	10000 s	159 µHz (500/Π) µHz	6283 s (2000*Π) s
1 GΩ (external)	10µF (with open amplifier)	10 µHz	100000 s	15.9 µHz (50/Π) µHz	62832 s (20000*Π) s

SINE GENERATOR



Another circuit for amplitude stabilization:



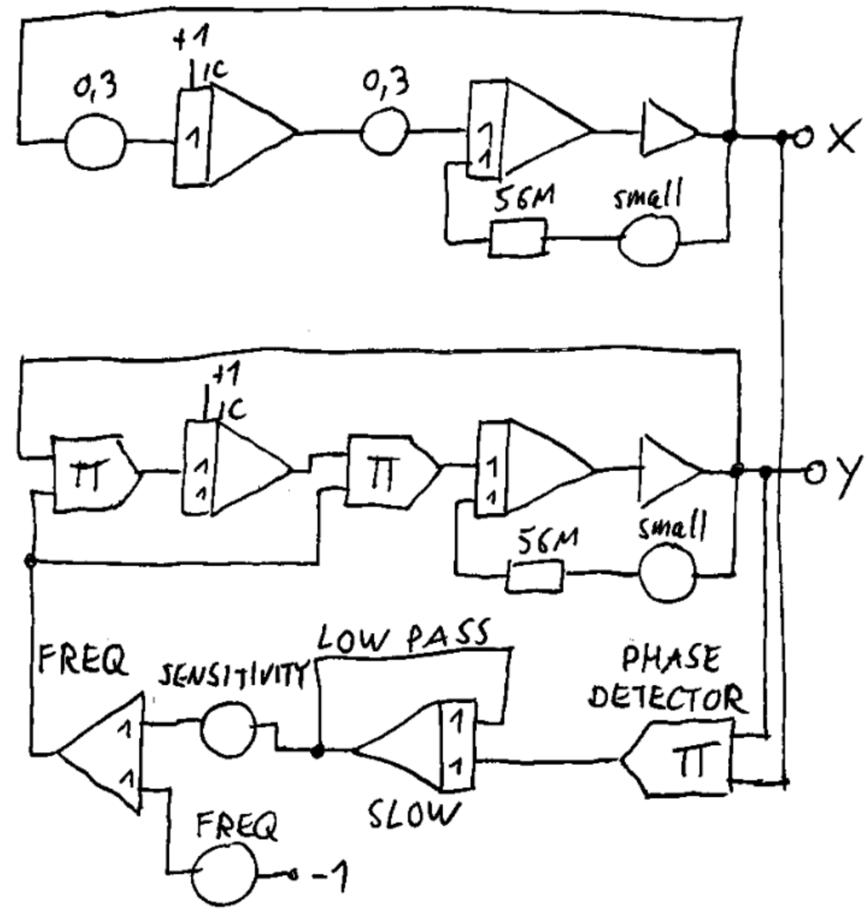
5.3 Phase Locked Loop

This circuit consists of two sine / cosine generators. The upper one has a fixed frequency and the lower one is adjustable by a control voltage.

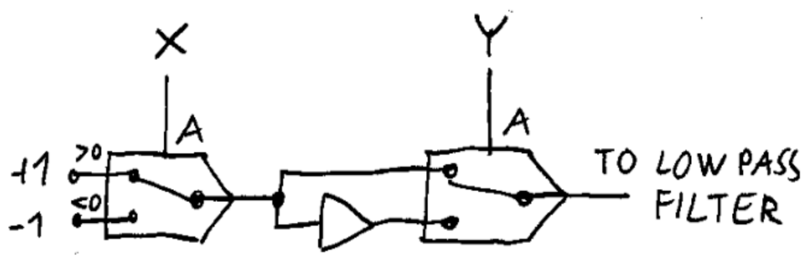
The question is how the two frequencies can be locked in an integer frequency ratio, for example 1:2.

A phase detector is required for a PLL (Phase Locked Loop). The simplest phase detector is an XOR gate, which is almost the same as a 4-quadrant multiplier. The output signal of the phase detector must be low-pass filtered, and then added to the control voltage.

Set SENSITIVITY to zero, tune the frequency to almost the correct frequency, then turn up SENSITIVITY. The second oscillator will lock to the first one. Look at the output signal of the low pass filter and then try to de-tune the frequency. Very interesting.

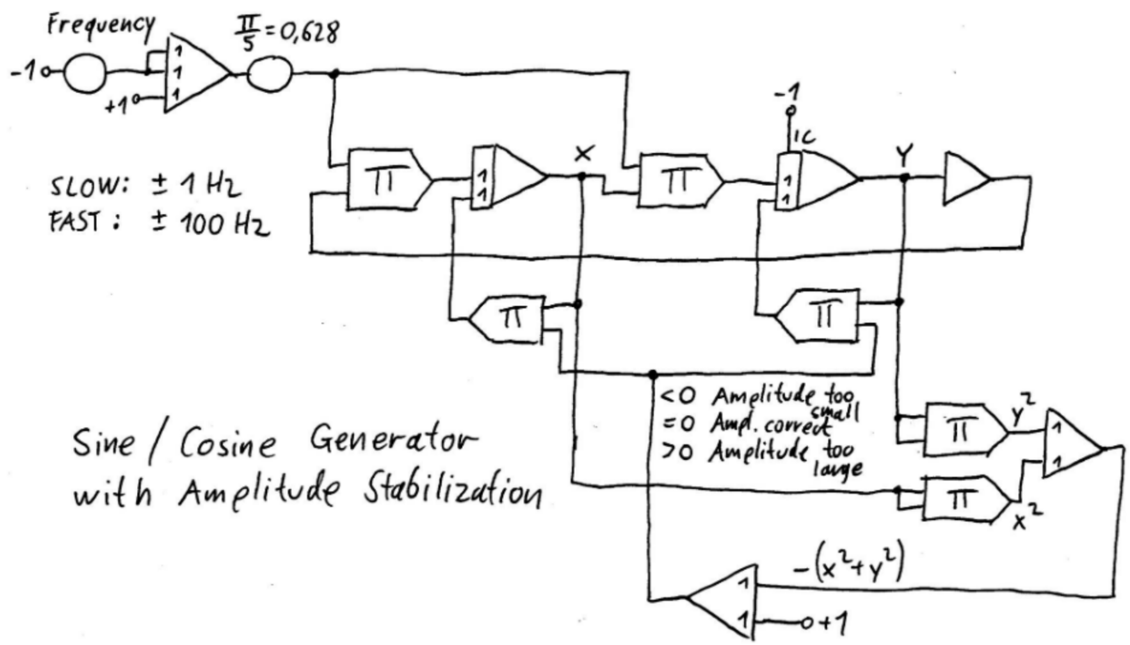


The phase detector can also be realized with two comparators. Then the whole circuit fits on one THAT.



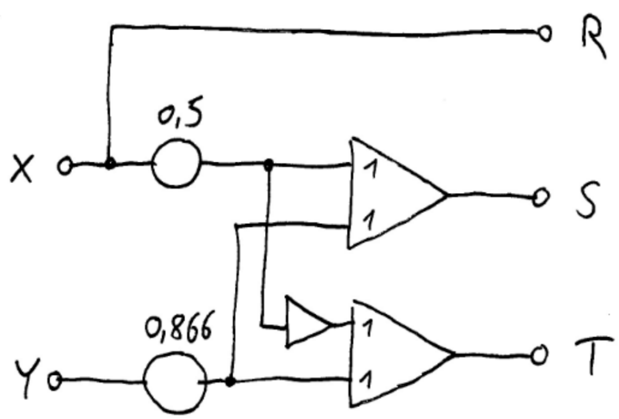
5.4 Sine / Cosine Generator with Amplitude Stabilization

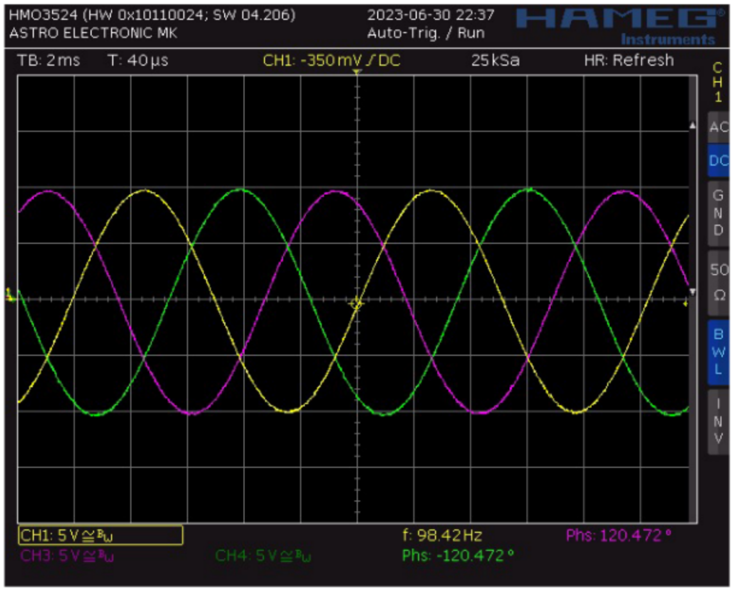
In analog sine / cosine generators the amplitude must somehow be stabilized. Two circuits with diodes are shown in Bernd's book "Analog and Hybrid Computer Programming" in figures 4.11 and 4.13. The drawback of amplitude stabilization with diodes is that it fails when the frequency becomes very small or zero.
In this circuit the stabilization does also work for small or zero frequency. The circuit in the lower right calculates the amplitude x^2+y^2 and compares it to the desired value 1. If an error is detected, it's corrected by the two multipliers in the center.
The drawback is that you need a total of 6 multipliers for this circuit.
This circuit can be further improved by making the amplitude corrections slower and allowing slightly larger error signals at the multiplier inputs, so that the multiplier's offset error doesn't produce extra errors. Insert two large resistors between the outputs of the error multipliers and the inputs of the integrators. 100 MΩ works fine, or even bigger resistor might be used.



5.5 3-Phase Generator

This circuit transforms the 90° phase shifted X and Y signals from the sine / cosine generator into three 120° phase shifted R, S, T signals. Works for all frequencies and for both turning directions.

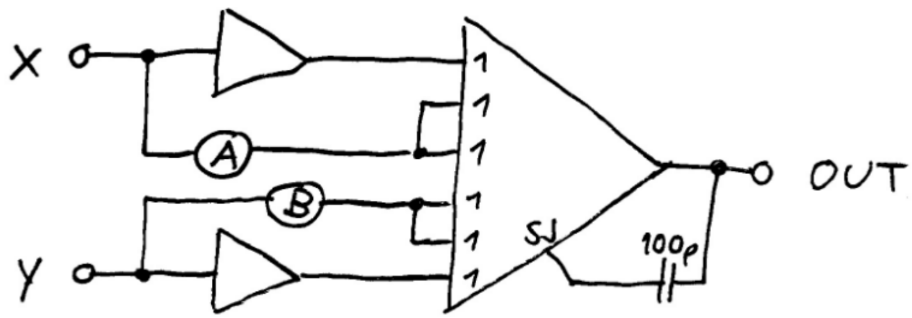




5.6 Arbitrary Phase Shift

This circuit transforms the 90° phase shifted X and Y signals from the sine / cosine generator into an output signal with phi degrees phase shift with respect to the X signal. Works for all frequencies and for both turning directions. If you see too much noise in the output, slow down the summer with the 100pF capacitor.
Because this summer has 6 inputs, a XIR block is required. I found out that the cable from SJ(Summer) to SJ(XIR) is receiving interference from other cables. That's why I had to slow down the summer with the 100pF capacitor. It helps to keep the SJ to SJ cable as short as possible, and away from other cables. Then 33pF is sufficient.

ARBITRARY PHASE SHIFT



$$A = 0{,}5\,(1 - \cos\varphi)$$
$$B = 0{,}5\,(1 - \sin\varphi)$$

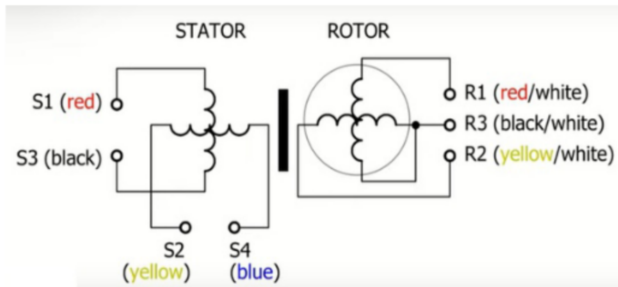
5.7 Resolvers

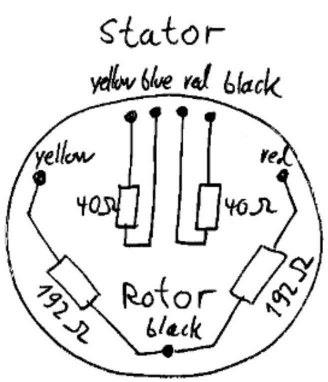
A resolver is an absolute angle encoder. The input and output signals are sine waves. https://en.wikipedia.org/wiki/Resolver_(electrical)

I did use this resolver which I found on Ebay:

Microtecnica Autosyn Resolver AY 528 A 45 A 2, manufactured under Bendix licence. Rotor 2 phase 26V Stator 2 phase 11.8V

The resistance of the two stator coils is 40 Ω each. With a 150 Ω resistor in series the total resistance is 190 Ω, the current is 15.8 mA at 3V (0.3 machine units). The rotor has two 90° phase shifted coils with 192 Ω each. One end of them is connected together.





For more informations about resolvers and synchros, see also:

https://www.analog.com/en/education/education-library/synchro-resolver-conversion.html

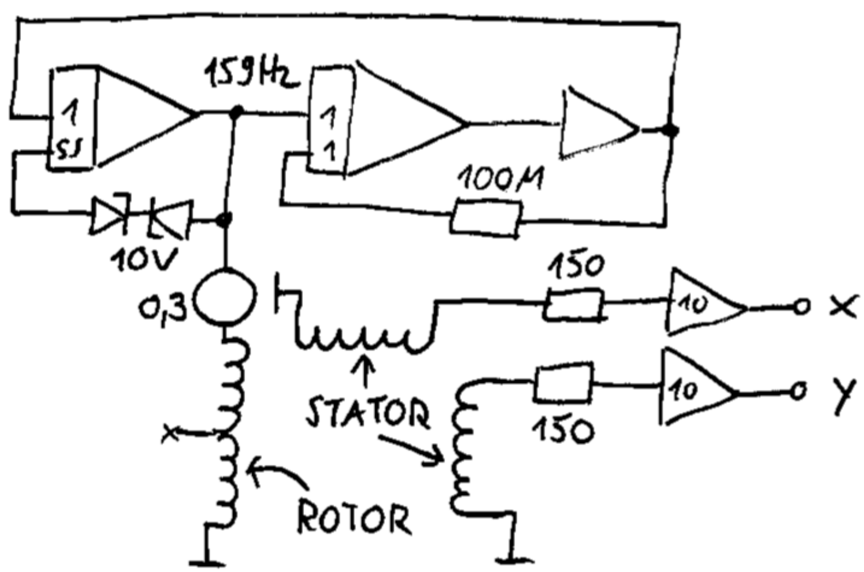
Resolvers have 2 phases with 90° phase shift, while synchros have 3 phases with 120° phase shift.

See also: https://analogmuseum.org/library/eai_handbook.pdf beginning at page 58

Nice video from Elektor: https://www.elektormagazine.com/news/resolver-synchro-introduction

Circuit 1:

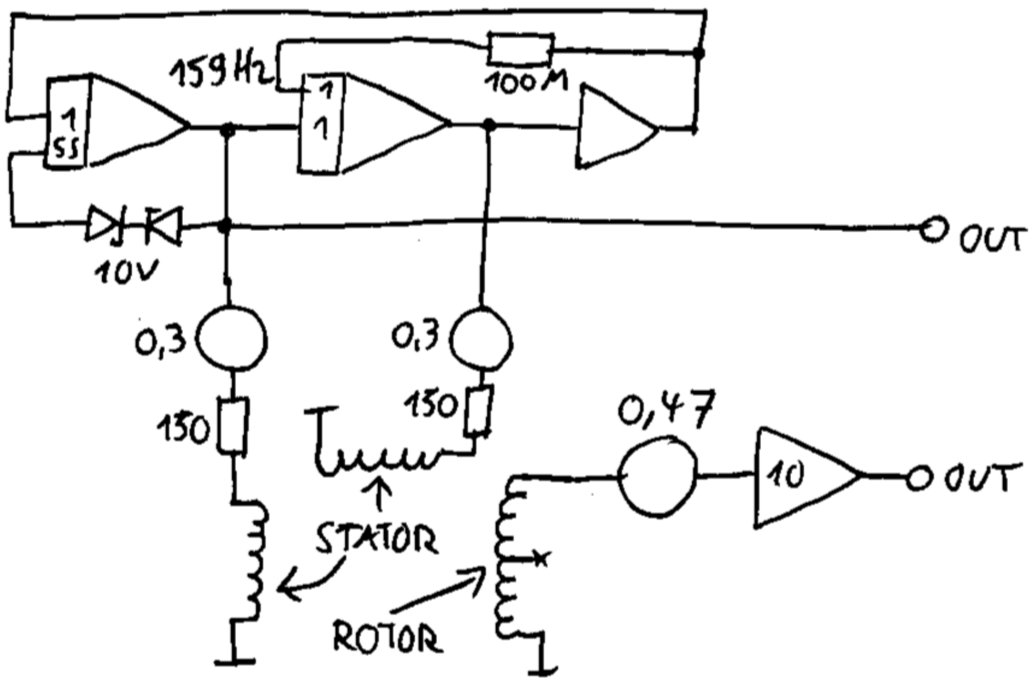
The input sine signal is connected to the rotor, and the voltages at the two stator coils are measured. The output signals are in phase and their amplitude and sign depends on the resolver's turning angle.



Alternatively it's also possible to connect the sine and cosine signals to the two rotor coils.

Circuit 2:

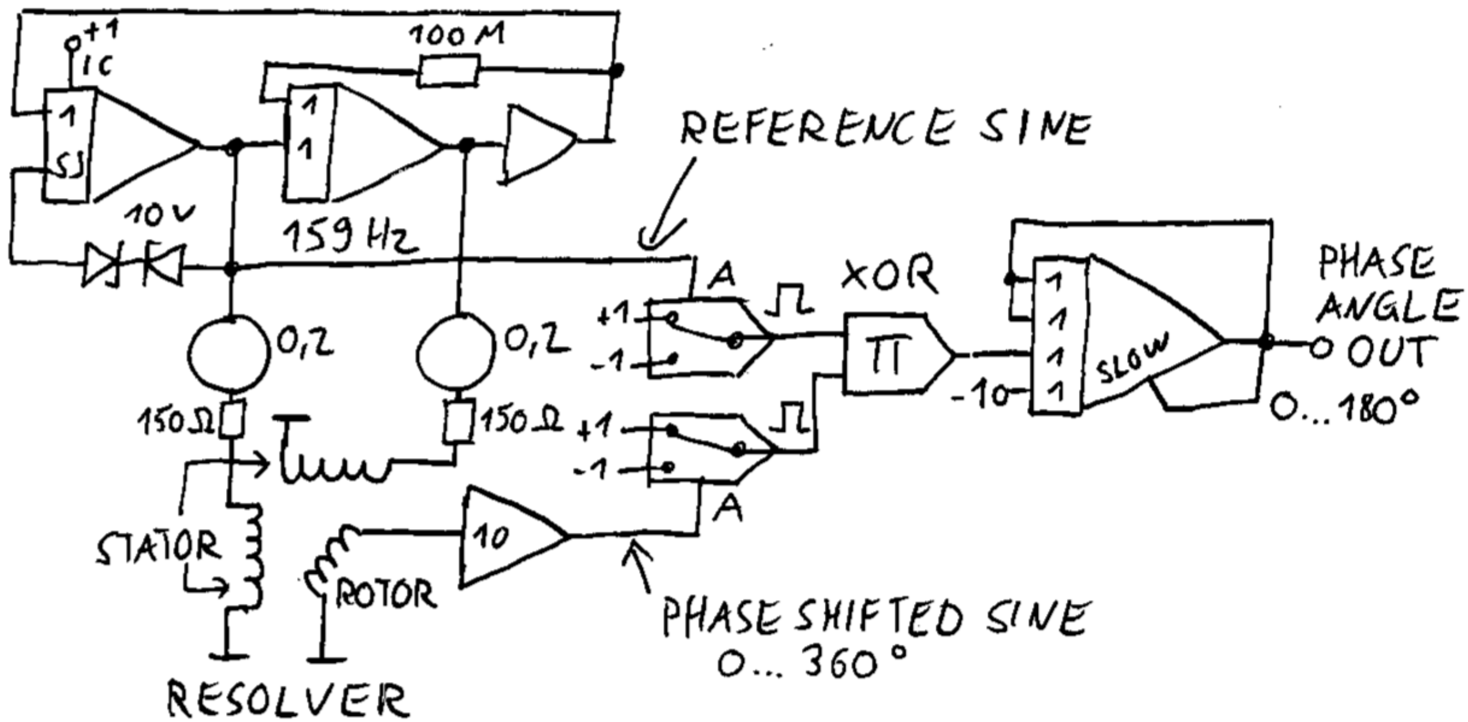
The input sine and cosine signals are connected to the stator coils, and the voltage at the rotor coils is measured. The output signal has constant amplitude and its phase shift with respect to the input signals depends on the resolver's turning angle.



Alternatively it's also possible to use both of the rotor coils, giving sine and cosine signals at the output.

5.8 Phase Angle Measurement

In this circuit the phase angle between two sine signals is measured. The reference signal is generated by a sine generator, and the phase shifted signal comes from a resolver. Both signals are converted to square waves and then a logical XOR is applied to them. If the logic levels are -1 and +1, the logical XOR can be realized with a multiplier. The result is shifted to the 0..1 range and low pass filtered.
Unfortunately the result is only from 0 to 180°, missing the sign of the phase shift. If you have an idea for +-180° output, please let me know.

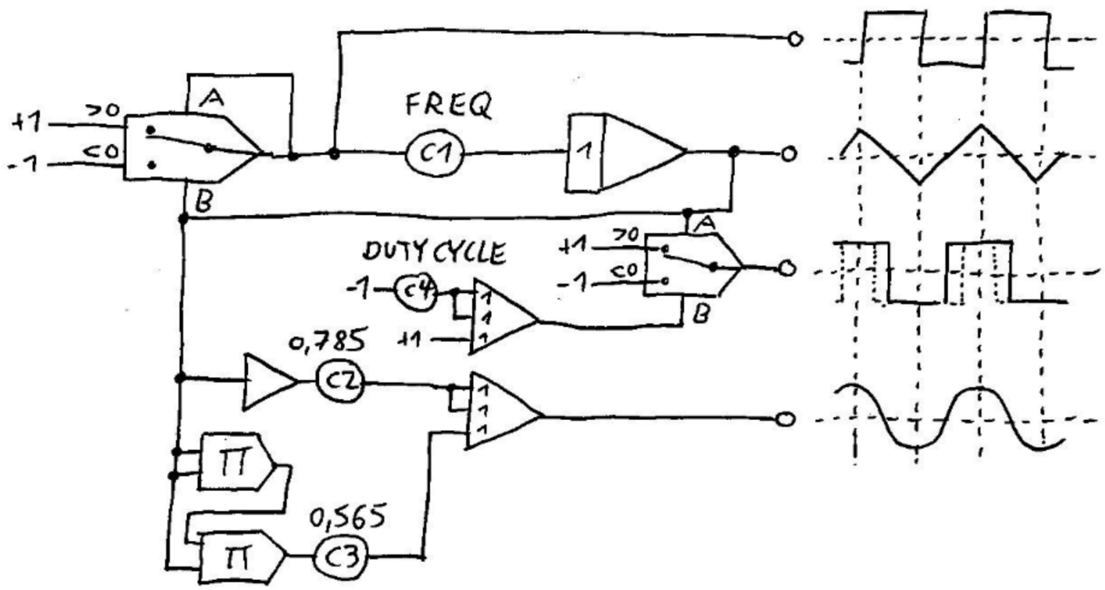


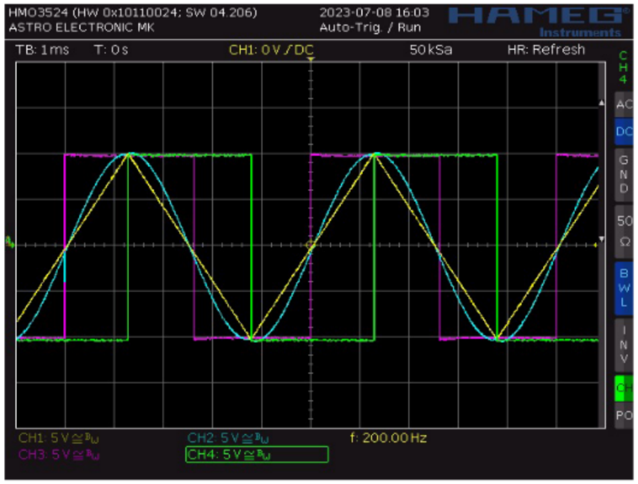
5.9 Approximated Sine Wave and Square Wave with adjustable Duty Cycle

This circuit does simultaneously generate a square wave, a triangular wave, a 90° phase shifted square wave with adjustable duty cycle, and an approximated sine wave. For zero slope at the min/max points the coefficients are C2 = 0.75 and C3 = 0.5.

I did also try to calculate the C2/C3 ratio from the power series of the sin() function, but the result doesn't look good, probably because the higher orders are missing. The zero slope approach is better.

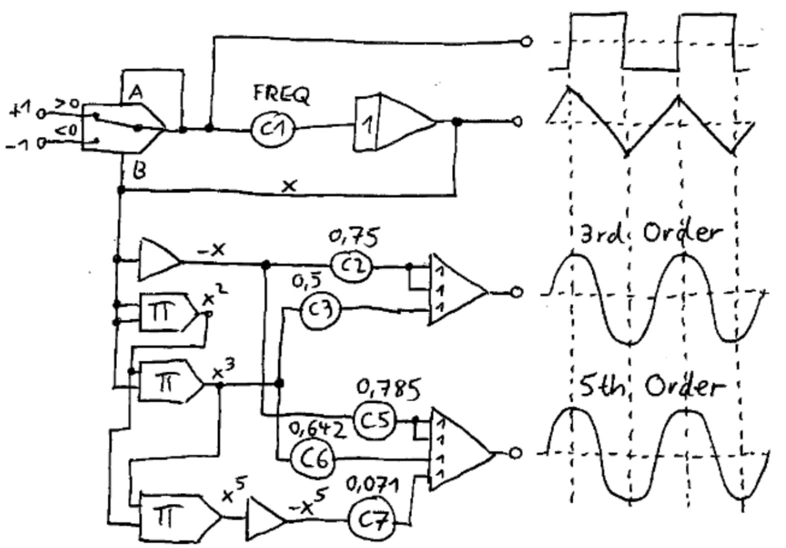
The frequency is f = 250Hz / (C1 * R), where R is the input resistor of the integrator in MΩ. The "1" input has 1 MΩ and the "10" input has 0.1MΩ.

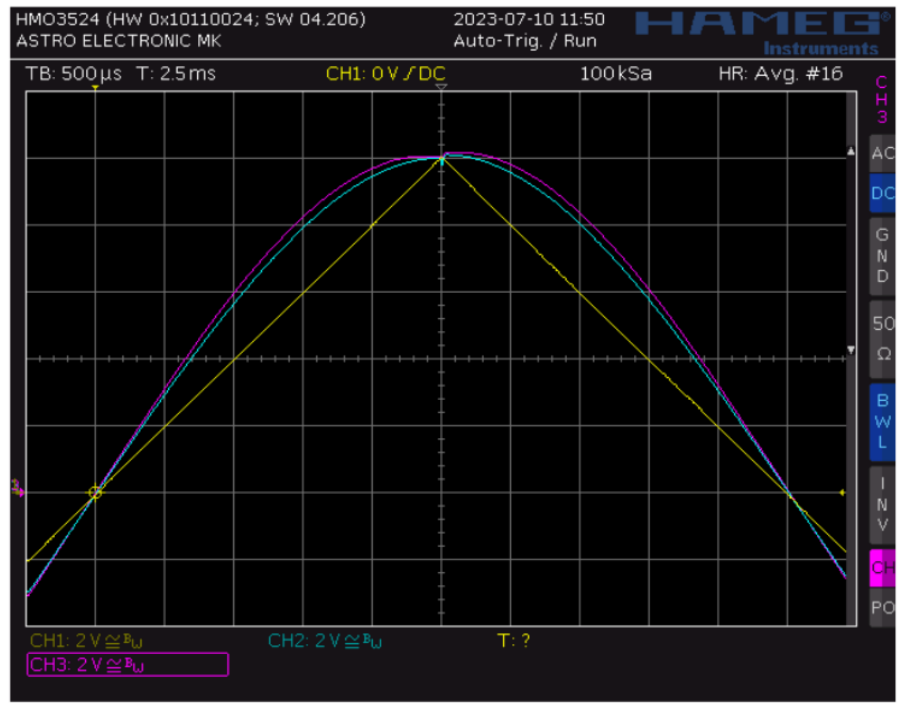




	3rd order sine approximation:	5th order sine approximation:
Equation and first derivative:	y = a * x + b * x^3 y' = a + 3b * x^2	y = a * x + b * x^3 + c * x^5 y' = a + 3b * x^2 + 5c * x^4
Condition 1: Peak amplitude = 1	y(1) = 1 a + b = 1	y(1) = 1 a + b + c = 1
Condition 2: Slope at peak = 0	y'(1) = 0 a + 3b = 0	y'(1) = 0 a + 3b + 5c = 0
Condition 3: Maximum slope = π/2	(only two conditions are possible)	y'(0) = π/2 a = π/2
Solution for coefficients:	a = 1.5 C2 = a/2 = 0.75 b = -0.5 C3 = -b = 0.5	a = π/2 = 1.571 b = 5/2-π = -0.642 c = (π-3)/2 = 0.071

Let's compare 3rd order (cyan) and 5th order (magenta) sine approximation. It's not a big difference:





The small step at the peak can be corrected by adding a very small amount of the square wave (from the comparator) to the approximated sine wave. Use a coefficient and a 100MΩ resistor.

See also: https://www.timstinchcombe.co.uk/index.php?pge=trisin

See also: Tietze, Schenk: Halbleiter-Schaltungstechnik, 11. Auflage, page 794

See also: Trigonometric Operations with the 433, in:

Analog Devices: The Best of Analog Dialogue, ISBN 0-9-916550-10-9, page 61 and 62

Using non-integer exponents:

sin(x) ≈ x - (x^2.827) / 6.28 (0 to π/2 radians) This has less than 0.25% error!

cos(x) ≈ 1 + 0.2325x - (x^1.504) / 1.445 (0 to π/2 radians) This has less than 1% error!

cos(x) = sin(π/2 - x) ≈ (π/2 - x) - ((π/2 - x)^2.827) / 6.28 (0 to π/2 radians) This has less than 0.25% error!

arctan(x) ≈ π/2 * x^1.2125 / (1 + x^1.2125)

Source: https://www.analog.com/media/en/analog-dialogue/volume-6/number-3/articles/volume6-number3.pdf#page=4

See also Bhāskara I's sine approximation formula: https://en.wikipedia.org/wiki/Bh%C4%81skara_I%27s_sine_approximation_formula

See also: https://www.analog.com/media/en/analog-dialogue/volume-18/number-1/articles/volume18-number1.pdf Page 14

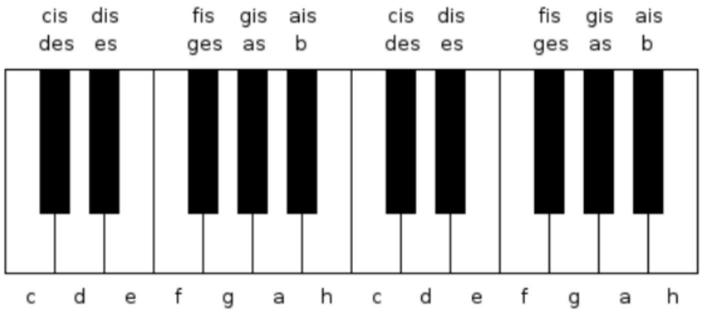
See also: Analog Devices: Nonlinear Circuits Handbook, Second Edition January 1976, page 60ff:

sin(x) ≈ 1.155 * x - 0.33 * x^2 +-2% error

sin(x) ≈ (1.0468 * x - 0.4278 * x^2) / (1 - 0.2618 * x) +-0.4% error

There are more approximations in the same chaptzer!

5.10 Keyboard with 1 V/Octave Signal, Triangle Wave Generator



If the integrator is used with an external input resistor connected to its SJ input, the resistor (in MΩ) can be calculated as follows:

R = 250 / f, where f is the frequency in Hz

The frequency can be increased by making the comparator limits smaller. Just insert a coefficient in the feedback from the comparator output to its input.

Frequency factor between two subsequent semitones: $2^{1/12} = 1.05946$

The resistors are calculated for a keyboard with 44 keys:

	c	cis/des	d	dis/es	e	f	fis/ges	g	gis/as	a	ais/b	h
Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
1						43.654 Hz 5.727 MΩ 5.257 MΩ	46.249 Hz 5.405 MΩ 4.935 MΩ	48.999 Hz 5.102 MΩ 4.632 MΩ	51.913 Hz 4.815 MΩ 4.345 MΩ	55.000 Hz 4.545 MΩ 4.075 MΩ	58.270 Hz 4.290 MΩ 3.820 MΩ	61.735 Hz 4.049 MΩ 3.579 MΩ
2	65.406 Hz 3.822 MΩ 3.352 MΩ	69.296 Hz 3.608 MΩ 3.138 MΩ	73.416 Hz 3.405 MΩ 2.935 MΩ	77.782 Hz 3.214 MΩ 2.744 MΩ	82.407 Hz 3.034 MΩ 2.564 MΩ	87.307 Hz 2.863 MΩ 2.393 MΩ	92.499 Hz 2.703 MΩ 2.233 MΩ	97.999 Hz 2.551 MΩ 2.081 MΩ	103.83 Hz 2.408 MΩ 1.938 MΩ	110.00 Hz 2.273 MΩ 1.803 MΩ	116.54 Hz 2.145 MΩ 1.675 MΩ	123.47 Hz 2.025 MΩ 1.555 MΩ
3	130.81 Hz 1.911 MΩ 1.441 MΩ	138.59 Hz 1.804 MΩ 1.334 MΩ	146.83 Hz 1.703 MΩ 1.233 MΩ	155.56 Hz 1.607 MΩ 1.137 MΩ	164.81 Hz 1.517 MΩ 1.047 MΩ	174.61 Hz 1.432 MΩ 961.6 kΩ	185.00 Hz 1.351 MΩ 881.3 kΩ	196.00 Hz 1.275 MΩ 805.4 kΩ	207.65 Hz 1.204 MΩ 733.9 kΩ	220.00 Hz 1.136 MΩ 666.3 kΩ	233.08 Hz 1.073 MΩ 602.5 kΩ	246.94 Hz 1.012 MΩ 542.3 kΩ
4	261.63 Hz 955.5 kΩ 485.5 kΩ	277.18 Hz 901.9 kΩ 431.9 kΩ	293.66 Hz 851.3 kΩ 381.3 kΩ	311.13 Hz 803.5 kΩ 333.5 kΩ	329.63 Hz 758.4 kΩ 288.4 kΩ	349.23 Hz 715.8 kΩ 245.8 kΩ	369.99 Hz 675.6 kΩ 205.6 kΩ	392.00 Hz 637.7 kΩ 167.7 kΩ	415.30 Hz 601.9 kΩ 131.9 kΩ	440.00 Hz 568.1 kΩ 98.15 kΩ	466.14 Hz 536.3 kΩ 66.26 kΩ	493.88 Hz 506.2 kΩ 36.16 kΩ
5	523.25 Hz 477.8 kΩ 7.75 kΩ											

First line: Frequency   Second line: Resistance R = 250 / f, where f is the frequency in Hz

Third line: Resistance minus 470 kΩ, because it makes sense to place a fixed 470 kΩ series resistor close to the summing junction.

See also: "Making music" https://analogparadigm.com/downloads/alpaca_27.pdf

Finally I decided that it's better to work with a keyboard that makes 1 V/Octave signal. The keyboard simply contains a passive voltage divider which is connected to the keys. Of course you are only allowed to press one key at a time.

1 V/Octave, the voltage step from one to the next semitone is 1/12 V = 0.0833 V.

	c	cis/des	d	dis/es	e	f	fis/ges	g	gis/as	a	ais/b	h
Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
2						87.307 Hz 2.417 V	92.499 Hz 2.5 V	97.999 Hz 2.583 V	103.83 Hz 2.667 V	110.00 Hz 2.75 V	116.54 Hz 2.833 V	123.47 Hz 2.917 V
3	130.81 Hz 3.00 V	138.59 Hz 3.083 V	146.83 Hz 3.167 V	155.56 Hz 3.25 V	164.81 Hz 3.333 V	174.61 Hz 3.417 V	185.00 Hz 3.5 V	196.00 Hz 3.583 V	207.65 Hz 3.667 V	220.00 Hz 3.75 V	233.08 Hz 3.833 V	246.94 Hz 3.917 V
4	261.63 Hz 4.00 V	277.18 Hz 4.083 V	293.66 Hz 4.167 V	311.13 Hz 4.25 V	329.63 Hz 4.333 V	349.23 Hz 4.417 V	369.99 Hz 4.5 V	392.00 Hz 4.583 V	415.30 Hz 4.667 V	440.00 Hz 4.75 V	466.14 Hz 4.833 V	493.88 Hz 4.917 V
5	523.25 Hz 5.00 V	554.37 Hz 5.083 V	587.33 Hz 5.167 V	622.25 Hz 5.25 V	659.26 Hz 5.333 V	698.46 Hz 5.417 V	739.99 Hz 5.5 V	783.99 Hz 5.583 V	830.61 Hz 5.667 V	880.00 Hz 5.75 V	932.33 Hz 5.833 V	987.77 Hz 5.917 V
6	1046.5 Hz 6.00 V											

Voltage divider for 10 V:

1131 Ω + 43 * 39 Ω + 1872 Ω = 4680 Ω   2.136 mA at 10V   2.4167 V + 43 * 0.0833 V + 4.0000 V = 10.0000 V

$U = 4.75\ V + \log_2(f / 440\ Hz)$ $\log_2(x) = \ln(x) / \ln(2)$ --> $U = 4.75\ V + \ln(f / 440\ Hz) / \ln(2)$ $U = 4.75\ V + 1.4427 * \ln(f / 440\ Hz)$

$f = 440\ Hz * 2^{(U - 4.75\ V)} = 16.3516\ Hz * 2^U$ In these formulas U is the control voltage in V

Frequency of triangle wave generator, where the "10" input of the integrator is used:

f = C * 2500 Hz, where C is the coefficient at the input of the integrator, or the control voltage at the inmput of a multiplier.

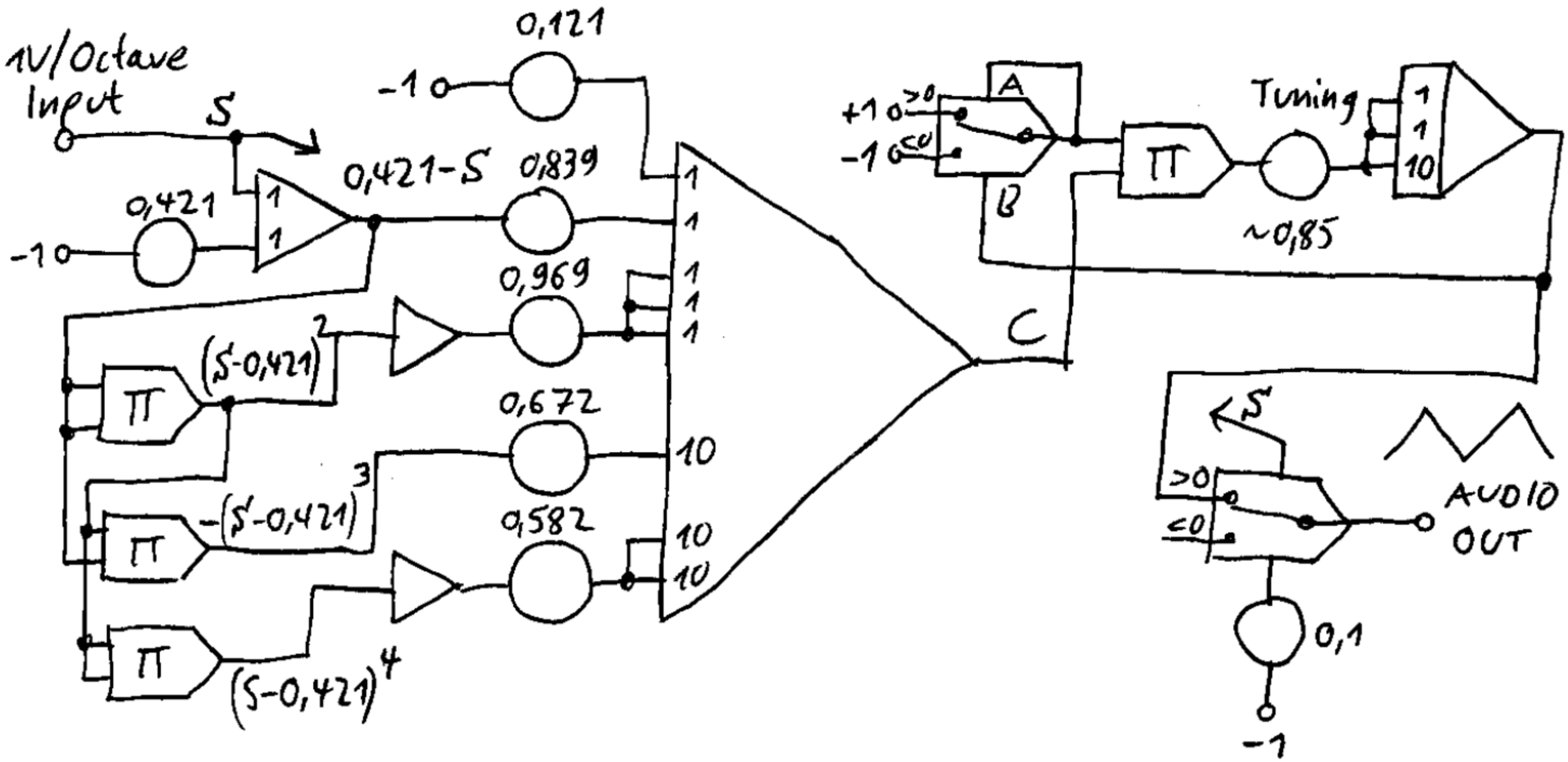
Calculate C as a function of the 1 V/octave control signal in machine units: S = U / 10

$C = 16.3516 * 2^{10S} / 2500 = 0.006541 * 2^{10S}$ S = 0.2417 ... 0.421 ... 0.6000   (0.421 is in the middle of the range)

Taylor series (from Wolfram Alpha): C = 0.121054 + 0.839085 (S - 0.421) + 2.90805 (S - 0.421)^2 + 6.71901 (S - 0.421)^3 + 11.6432 (S - 0.421)^4

Use this text as input for Wolfram Alpha: "series of y=0.006541*2^(10*x) at x=0.421"

This circuit gets a 1 V/Octave input signal from a keyboard and makes a triangular wave. This is version 1: Not good, don't use it !



My measurement results (the oscillator was calibrated at 440Hz):

Note	Frequency Hz	Control voltage S		C (calculated) in machine units				Frequency (calculated) (Error > 1%) (Error > 2%)			Frequency (measured) (Error > 1%) (Error > 2%)			
		Volt	Machine units	2^{nd} order	3^{rd} order	4^{th} order	exact	2^{nd} order Hz	3^{rd} order Hz	4^{th} order Hz	Analog 4^{th} order, Version 1 Hz	Analog 4^{th} order, Version 2 Hz	Analog 4^{th} order, Version 3 Hz	Digital 12-bit look-up-table in Teensy LC Hz
F2	87.307	2.417	0.2417	0.0641	0.0254	0.0374	0.0349	160.24	63.414	93.498	106	96.2	92.9	86.8
A2	110.00	2.75	0.275	0.0605	0.0396	0.0449	0.0440	151.34	99.063	112.29	124	114.8	112.0	109.7
C3	130.81	3.00	0.300	0.0621	0.0502	0.0527	0.0523	155.25	125.50	131.74	143	134.1	131.9	130.7
F3	174.61	3.417	0.3417	0.0728	0.0695	0.0699	0.0698	182.00	173.63	174.78	184	177.2	176.0	175.2
A3	220.00	3.75	0.375	0.0886	0.0880	0.0880	0.0880	221.52	219.89	220.02	228	222.4	221.9	220.9
C4	261.63	4.00	0.400	0.1047	0.1047	0.1047	0.1047	261.79	261.63	261.64	269	263.5	263.5	262.4
F4	349.23	4.417	0.4417	0.1397	0.1397	0.1397	0.1397	349.17	349.32	349.33	354	350.0	350.2	349.7
A4	440.00	4.75	0.475	0.1748	0.1759	0.1760	0.1760	437.11	439.76	440.00	440	440.0	440.0	440.0
C5	523.25	5.00	0.500	0.2055	0.2088	0.2093	0.2093	513.73	522.01	523.14	519	520.6	520.4	522.9
F5	698.46	5.417	0.5417	0.2647	0.2765	0.2790	0.2794	661.74	691.28	697.46	683	688.8	688.2	694.4
A5	880.00	5.75	0.575	0.3192	0.3438	0.3503	0.3520	798.10	859.45	875.82	850	859.1	858.5	872.2
C6	1046.5	6.00	0.600	0.3644	0.4030	0.4149	0.4186	911.07	1007.41	1037.29	1000	1010	1010	1032

Although the errors of the Taylor series are already quite large, most of the errors in version 1 came from inaccuracies in the analog computation. Let's analyze what's the problem. The term (S - 0.421) is always in the range from -0.1793 to +0.179. If this variable is raised to a power, it becomes very small. It becomes so small, that it's domonated by the offset errors of the analog multipliers. And then the variable (and the error) is multiplied by quite large coefficients, the largest one is 11.643. This Taylor series is mathematically correct, but it's not optimized for the analog computer!

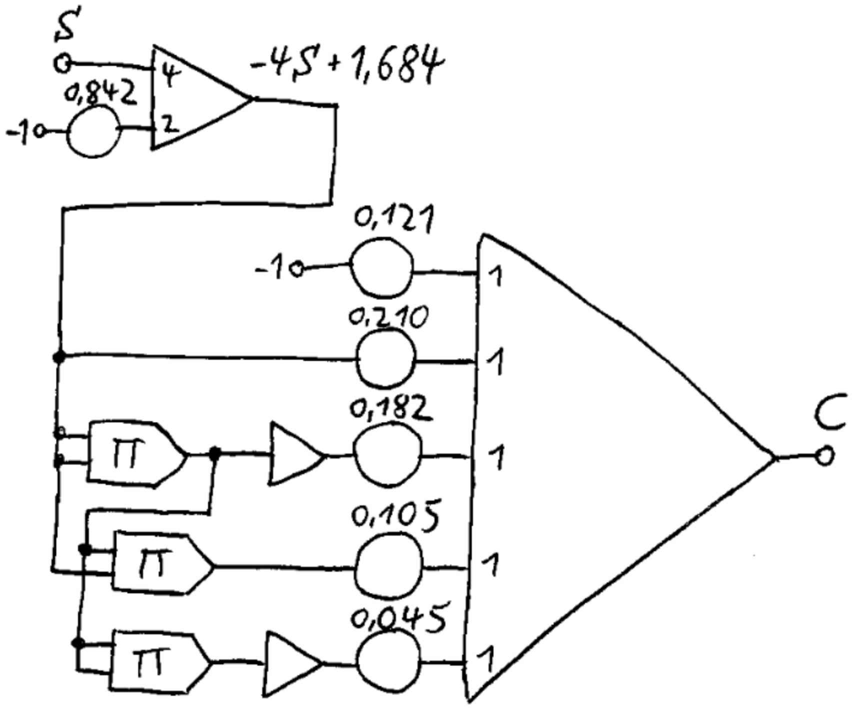
C = 0.121 + 0.839 (S - 0.421) + 2.908 (S - 0.421)^2 + 6.719 (S - 0.421)^3 + 11.643 (S - 0.421)^4 Version 1, not optimized for analog computer !

How can the Taylor series be optimized?

Let's multiply the term (S - 0.421) by 4 (even 5 would have been possible). Then it's always in the range from -0.7172 to +0.716. When this variable is raised to the 4^{th} power, it's still 0.26. To compensate for this change, we must divide the coefficient in front of the term by 4 raised to the corresponding power. Now we have this Taylor series, which is mathematically exactly the same as before, but it's much less sensitive to offset errors in the multipliers:

C = 0.121 + 0.210 * (4S - 1.684) + 0.182 (4S - 1.684)^2 + 0.105 (4S - 1.684)^3 + 0.045 (4S - 1.684)^4 Version 2, optimized for analog computer !

This is the circuit for the improved version 2:



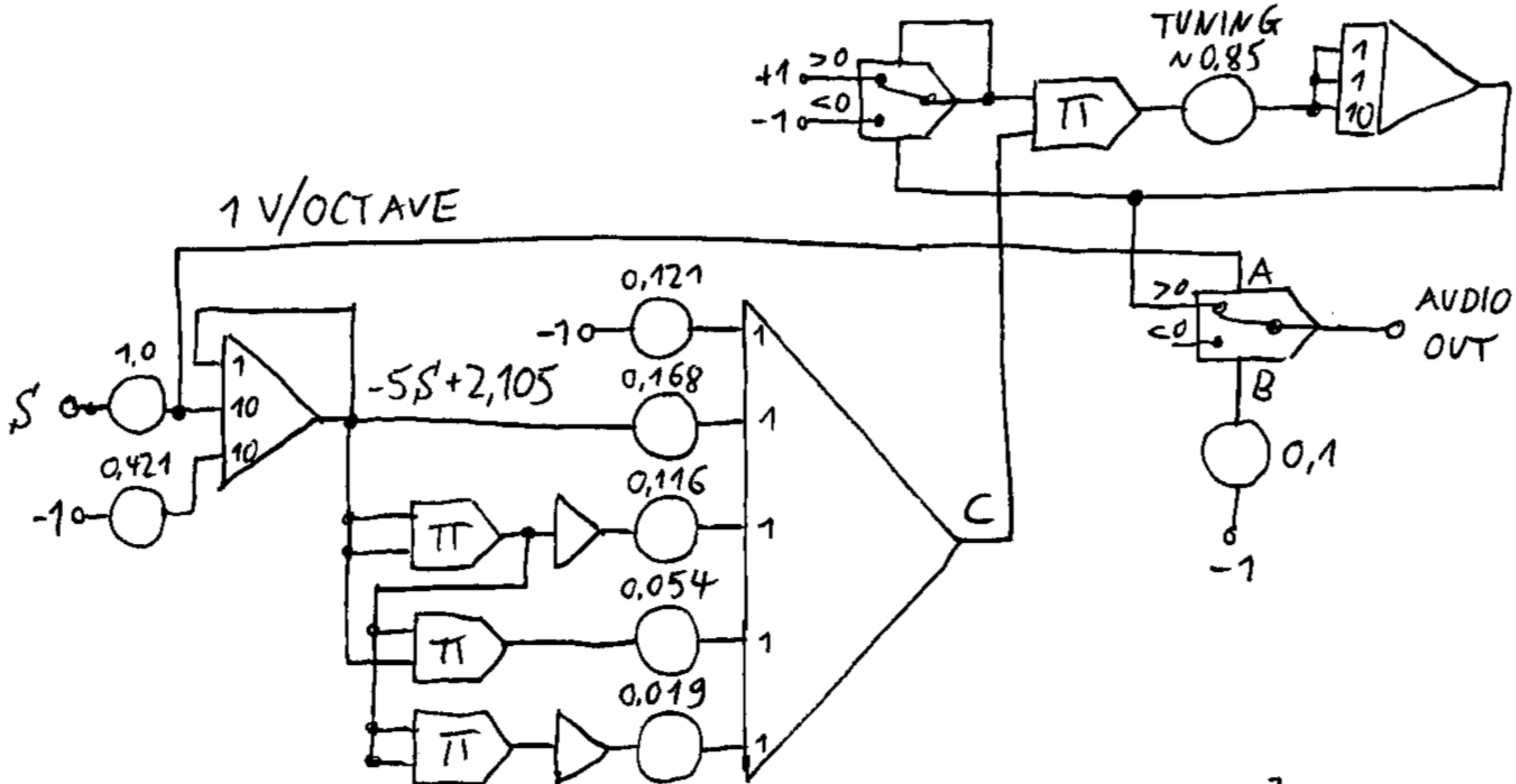
Can it be further improved? Yes, there are two things that can be improved:

First, the term (S - 0.421) can be multiplied by 5 instead of 4. The result still fits in the -1 to +1 machine unit range.

Second, the input signal comes from a passive resistor divider in the keyboard, which is not a low impedance source. It should go to a high impedance input. In version 2 it did go to 5 inputs with 1MΩ input resistance each, so the total input impedance was 200kΩ. This can be improved by inserting a coefficient which is set to 1.000. Then the input impedance is 1MΩ.

C = 0.121 + 0.168 * (5S - 2.105) + 0.116 (5S - 2.105)^2 + 0.054 (5S - 2.105)^3 + 0.019 (5S - 2.105)^4 **Version 3, optimized for analog computer !**

This is version 3:



$$C = 0{,}121 + 0{,}168(5S'-2{,}105) + 0{,}116(5S'-2{,}105)^2 + 0{,}054(5S'-2{,}105)^3 + 0{,}019(5S'-2{,}105)^4$$

For the last column the function C = f(S) was realized as a digital look-up-table in Teensy LC, with 12-bit input and output. A 19th order Taylor series was used as an approximation for the exponential function, because the built-in exp() function needs too much memory:

```
float function(float x)   // This is the user-defined function with input and output in machine units
{
  return(0.006541 * expo(6.931471806 * x));
}
```

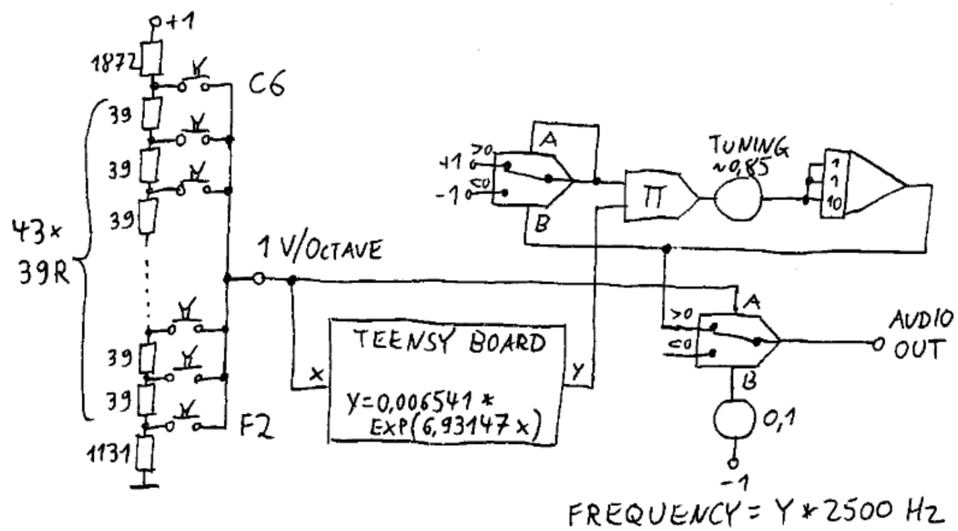
```
float expo(float x)    // This is a power series approximation for the exponential function,
{                      // because the built-in exponential function needs too much RAM
  float y = 1;
  float z = 1;
  float n = 1;
  for (int i = 1; i < 20; i++)
  {
    z = z * x;
    n = n * i;
    y = y + z / n;
  }
  return(y);
}
```

$C = 0.006541 * 2^{10S}$

$= 0.006541 * \exp(10 * x * \ln(2))$

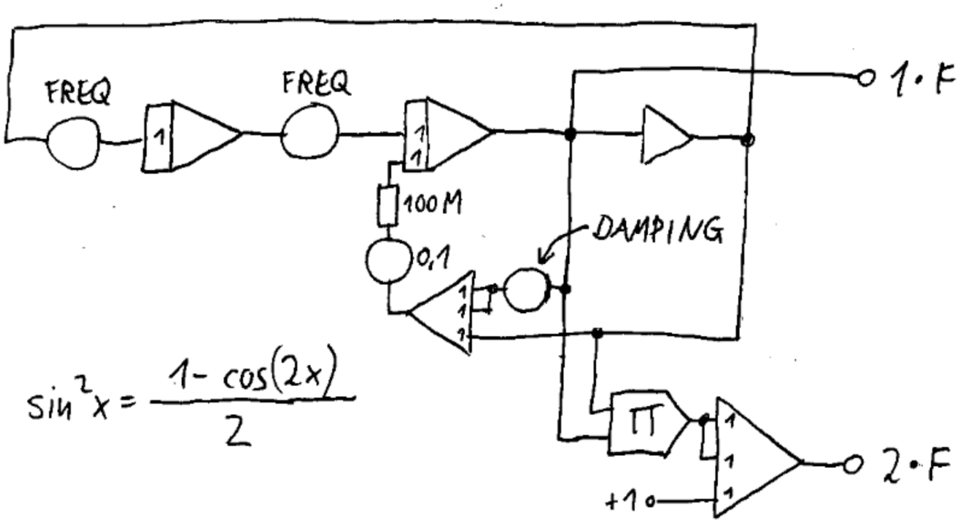
$= 0.006541 * \exp(6.931471806 * x)$

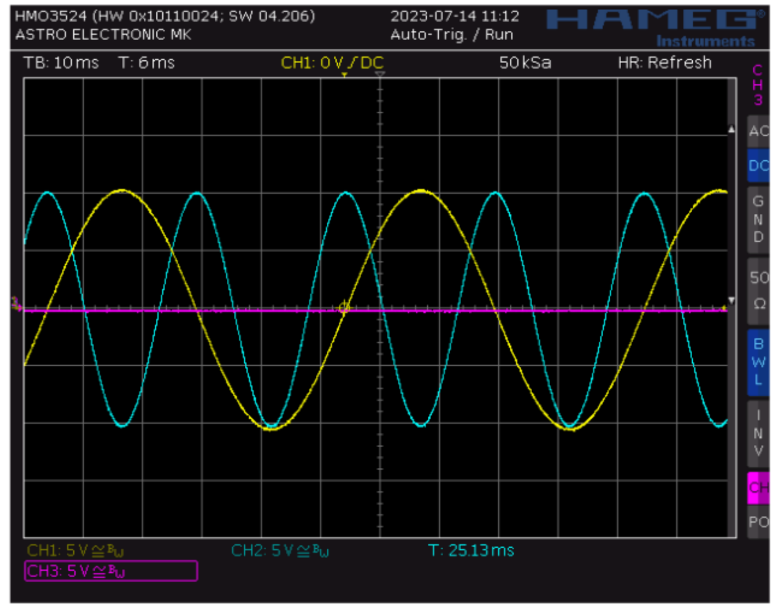
This is the circuit with the Teensy LC board as a look-up-table:



5.11 Frequency Doubling for Sine Waves

Forgot the initial condition for one integrator in the schematic.



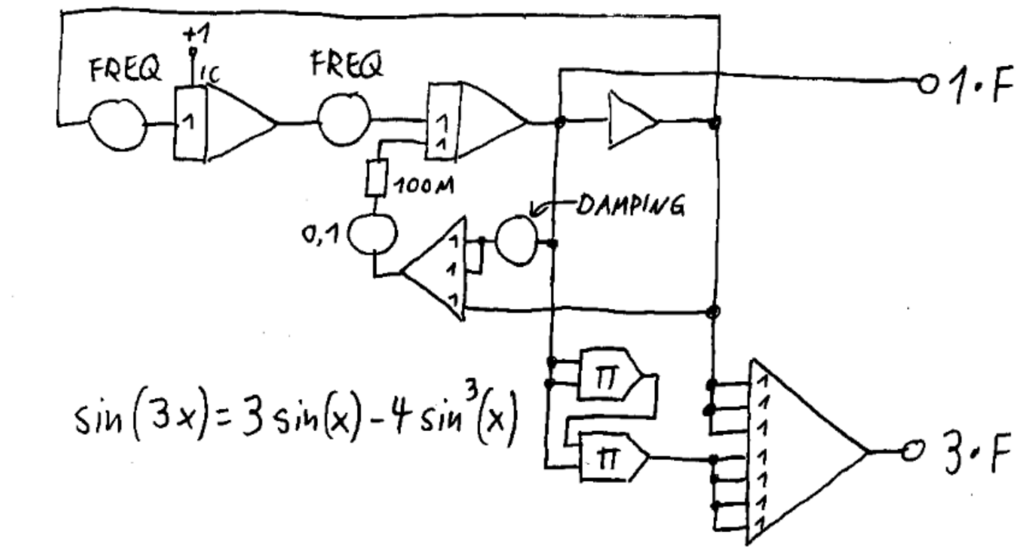


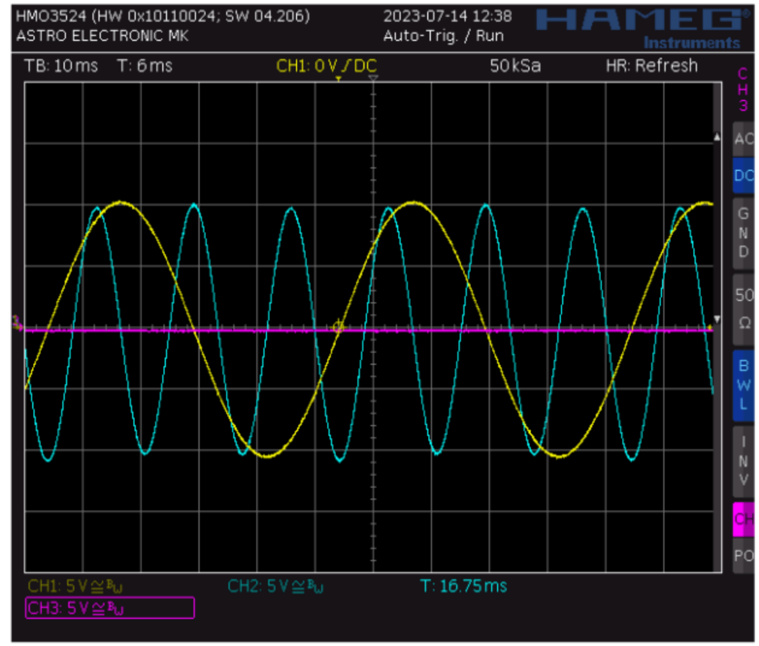
It's possible to use one of these formulas:

cos(2x) = 1 - 2 * sin²(x)

sin(2x) = 2 * sin(x) * cos(x)

5.12 Frequency Tripling for Sine Waves





It's possible to use one of these formulas:

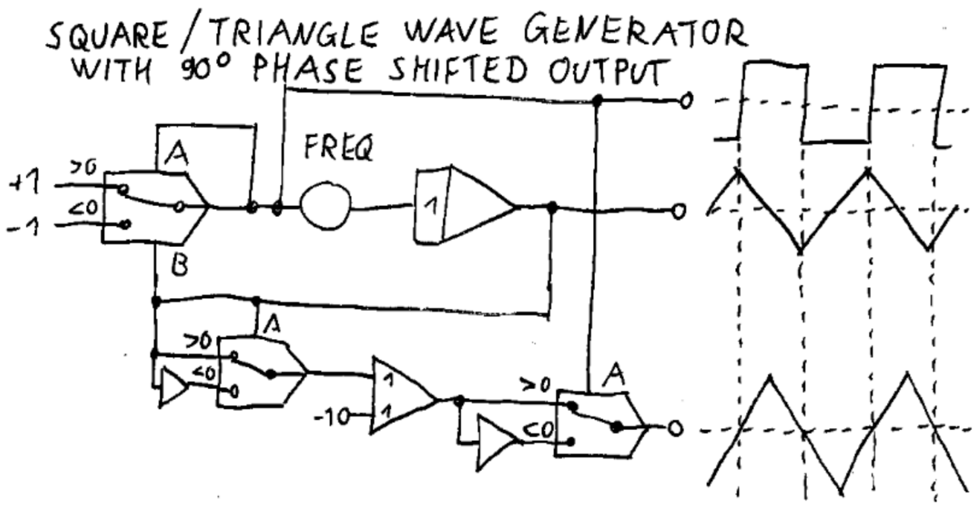
$\sin(3x) = \sin(x) * (1 + 2\cos(2x))$

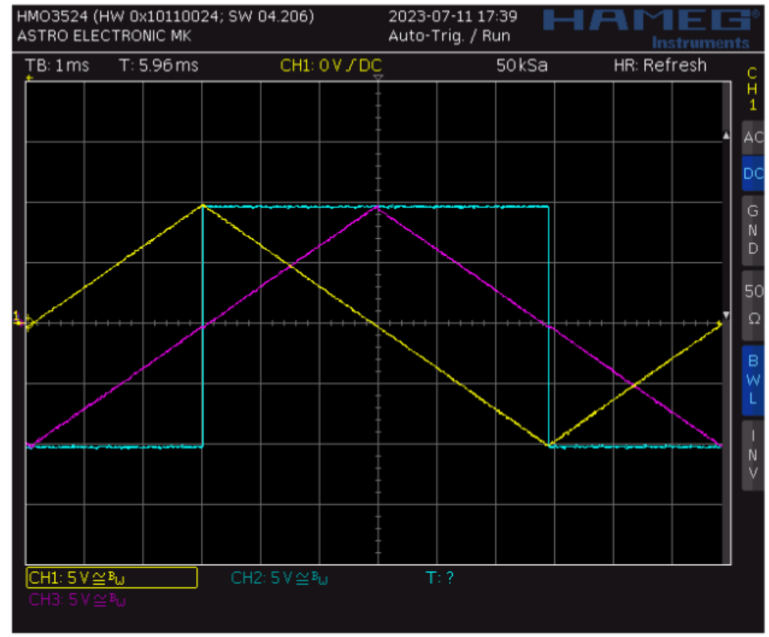
$\sin(3x) = 3 * \sin(x) - 4 * \sin^3(x)$

$\cos(3x) = \cos(x) * (2 * \cos(2x) - 1)$

$\cos(3x) = 4 * \cos^3(x) - 3 * \cos(x)$

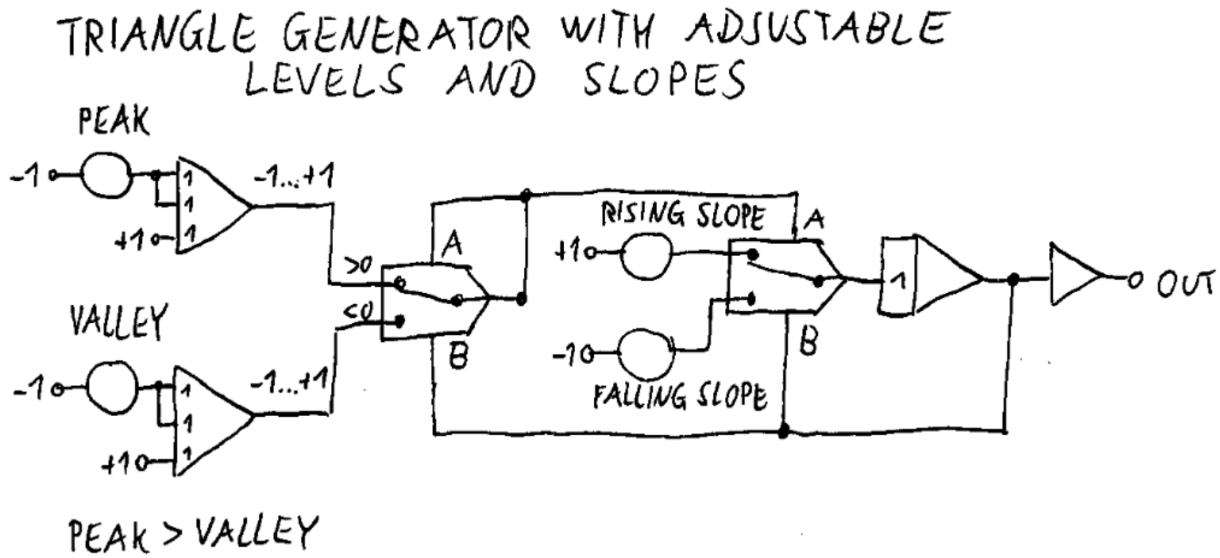
5.13 Square and Triangle Wave Generator with 90° Phase Shift



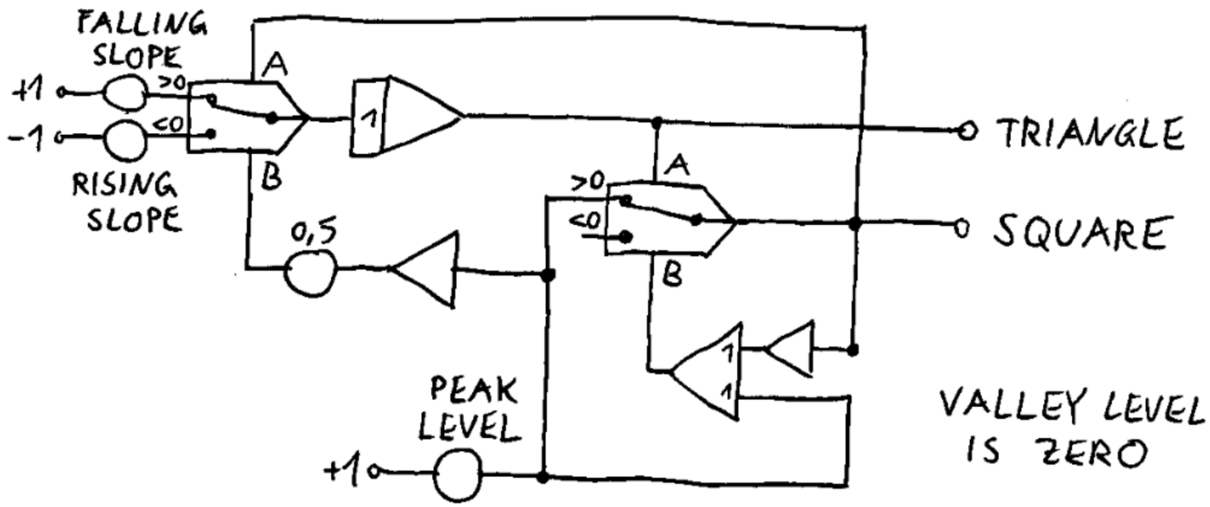


5.14 Triangle Generator with adjustable Levels and Slopes

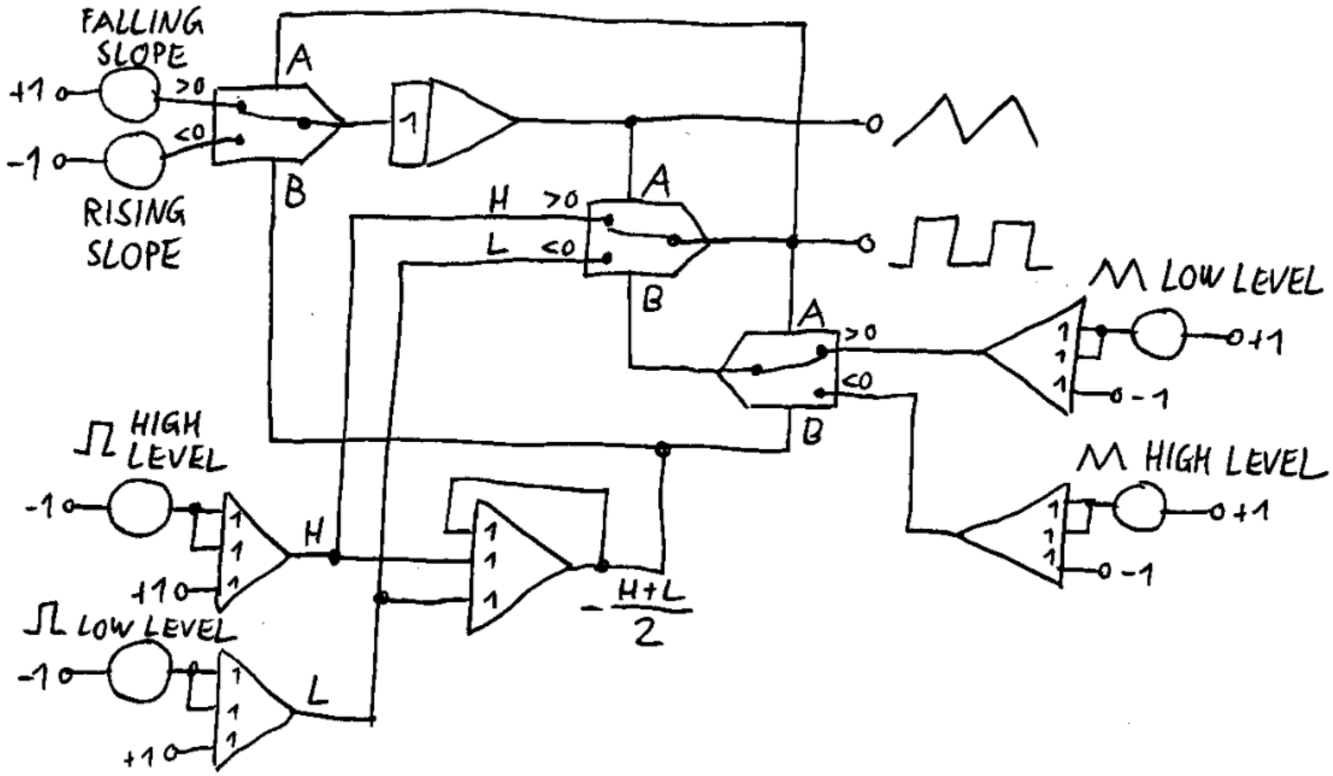
Four parameters can be adjusted: Peak level, valley level, rising slope and falling slope. Peak level must not be smaller than valley level. Peak and valley can also be both positive or both negative.



This triangle / square wave generator has an adjustable peak level (for both waveforms) and the valley level is zero (for both waveforms):



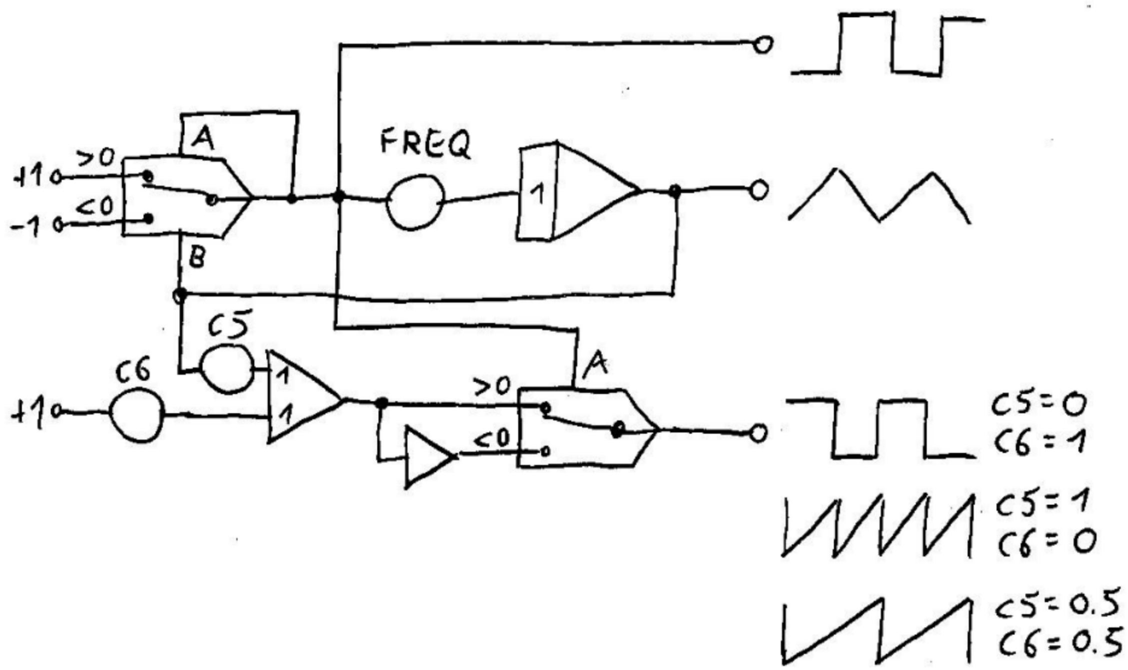
In this triangle / square wave generator everything is adjustable:



5.15 Sawtooth Generator

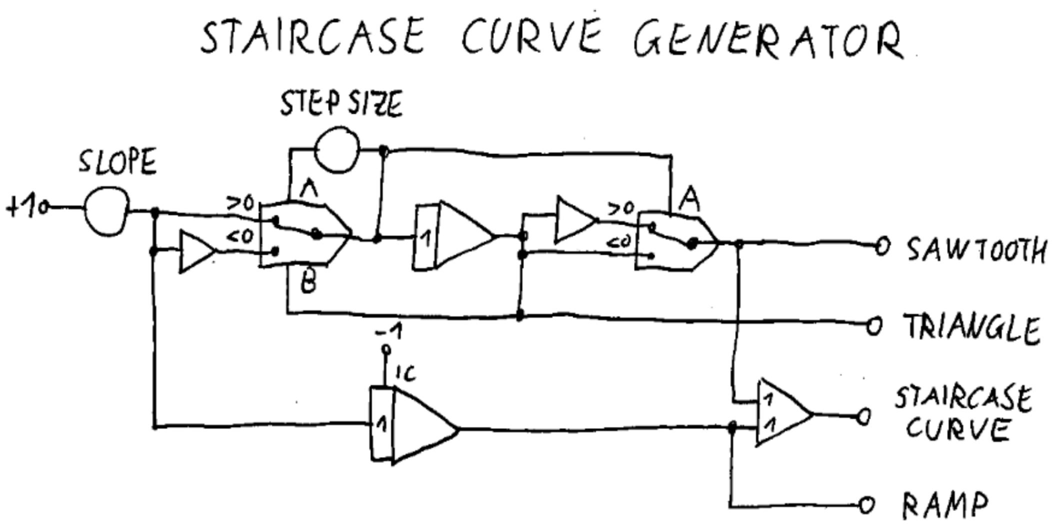
The upper half of this circuit is the well known triangle / square wave generator from figure 5.13 in Bernd's book. The output of the lower half depends on the setting of coefficients C5 and C6, and can be a square wave, or a sawtooth wave with the same or with double frequency.

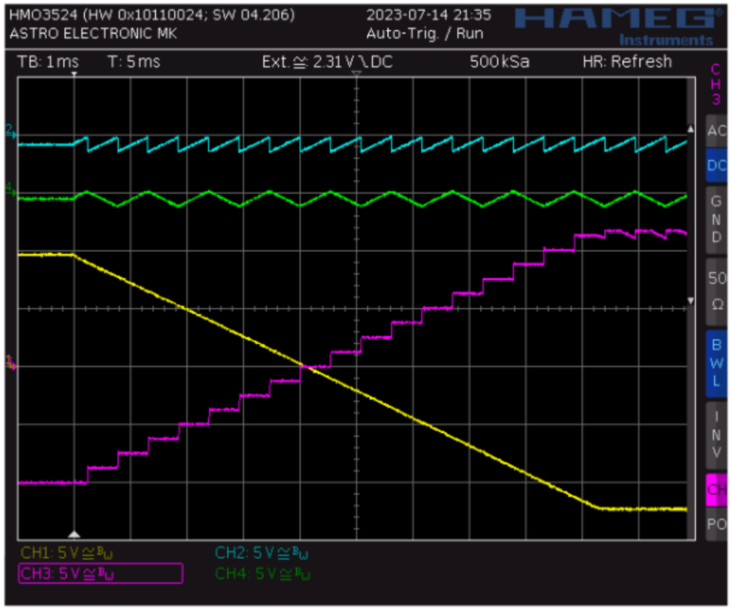
If you want falling sawtooth instead of rising sawtooth, you can swap the inputs of the second comparator.



5.16 Staircase Curve Generator

A staircase curve is simply the sum of a ramp signal and a sawtooth signal, which both have the same slope but with opposite sign. Make sure that the two integrators have the same time constant.

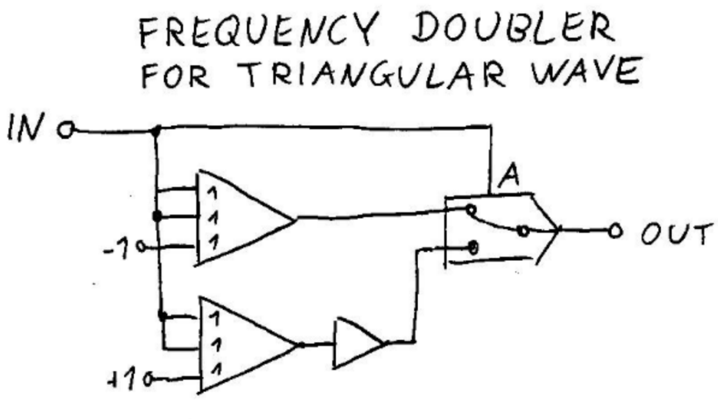


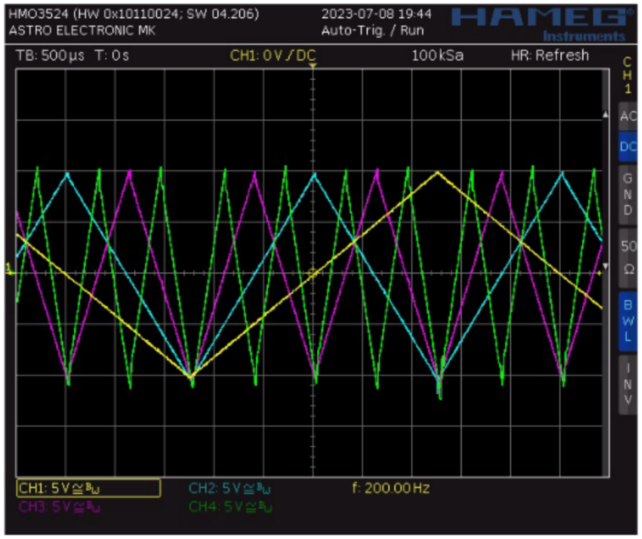


Note: It might be required to give the upper integrator a small non-zero initial condition.

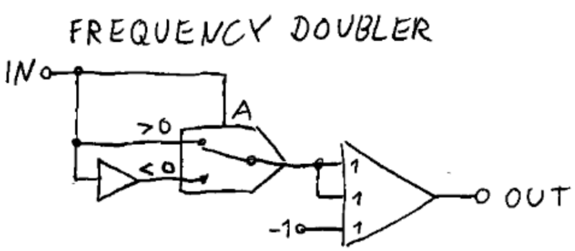
5.17 Frequency Doubling for Triangular Waves

This is a frequency doubler for triangular waves. It does only work if the input signal has the correct amplitude from -1 to +1. The output can be cascaded to the input of the next doubler stage. In the oscilloscope screenshot yellow is the input signal, and cyan, magenta and green are cascaded outputs. This example runs on two THATs.

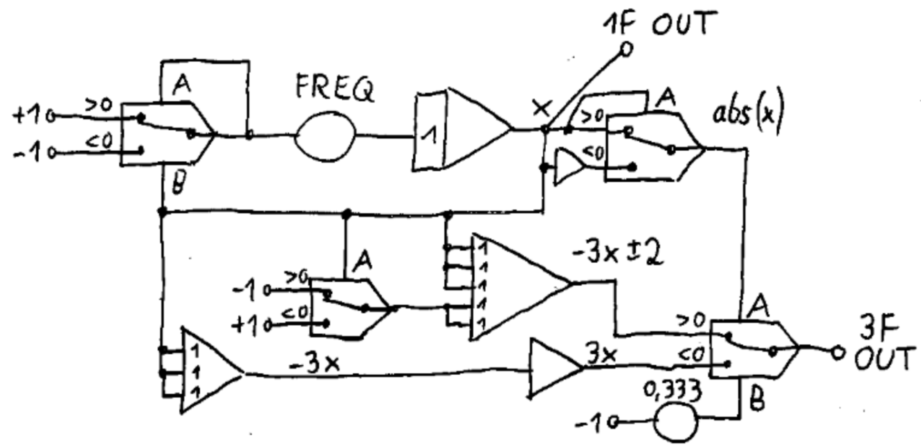


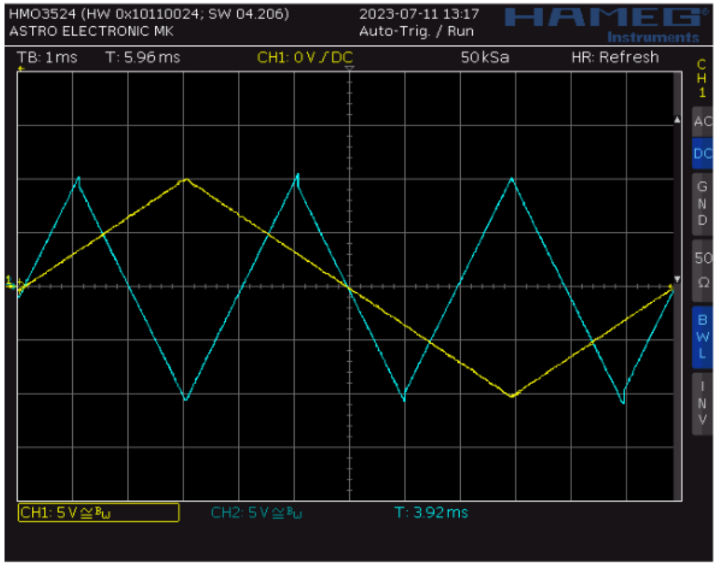


This circuit is simpler and has the same function:

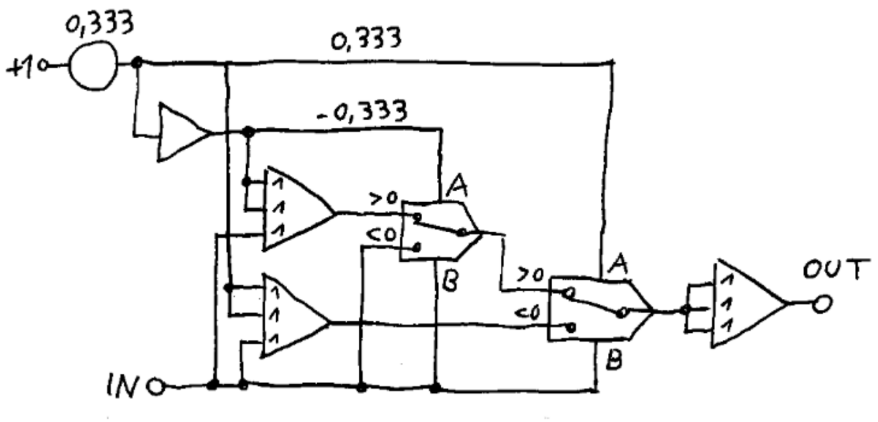


5.18 Frequency Tripling for Triangular Waves





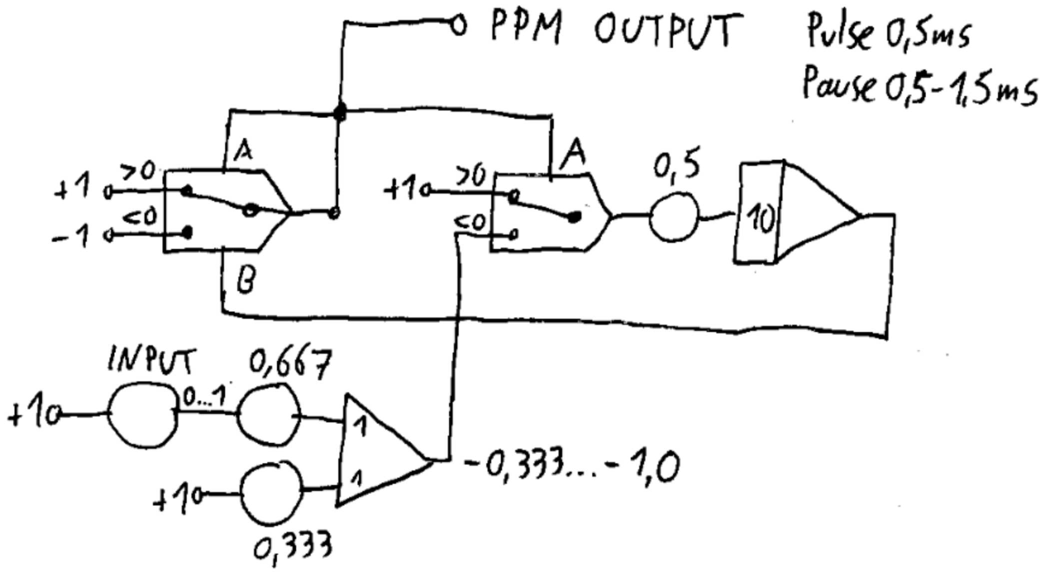
This frequency tripling circuit is better (output is inverted):



5.19 Pulse Position Modulation

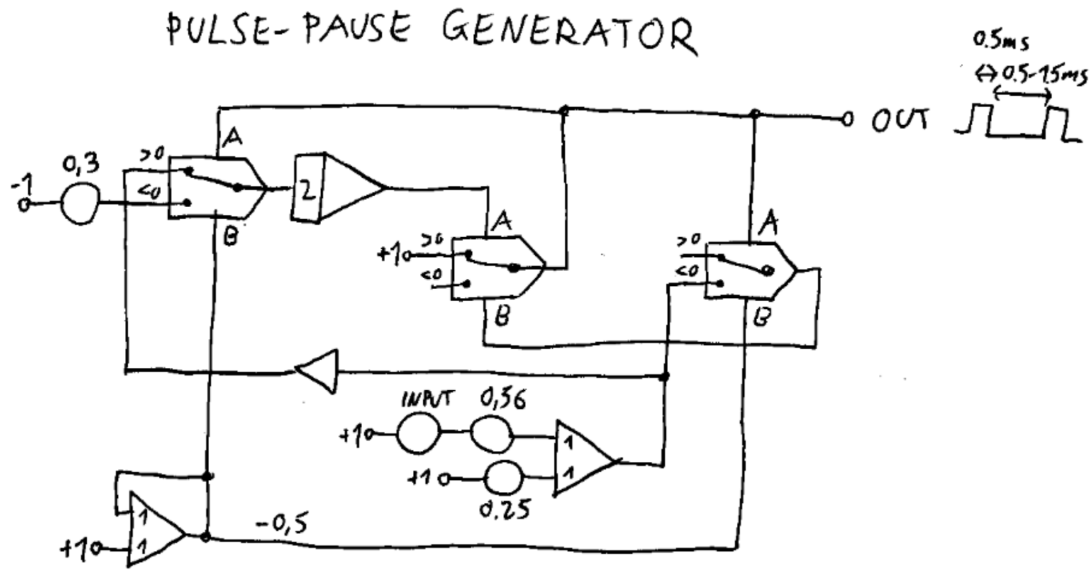
Simple example for pulse position (or pulse pause) modulation. It has a fixed 0.5ms pulse width and a variable 0.5ms - 1.5ms pause. But the pause width is a nonlinear function of input voltage. Who has a better idea to make it linear?

We need a voltage to time converter. The problem is that this circuit is a voltage to slope converter. Slope is voltage / time. That's why in this circuit the voltage is proportional to 1/time. I think the trick is to charge the capacitor with a constant slope to a variable voltage. Then time is proportional to voltage. But how can the capacitor (with variable voltage) be discharged in constant 0.5ms?



This is the solution. The pause duration does linearly depend on the input signal.

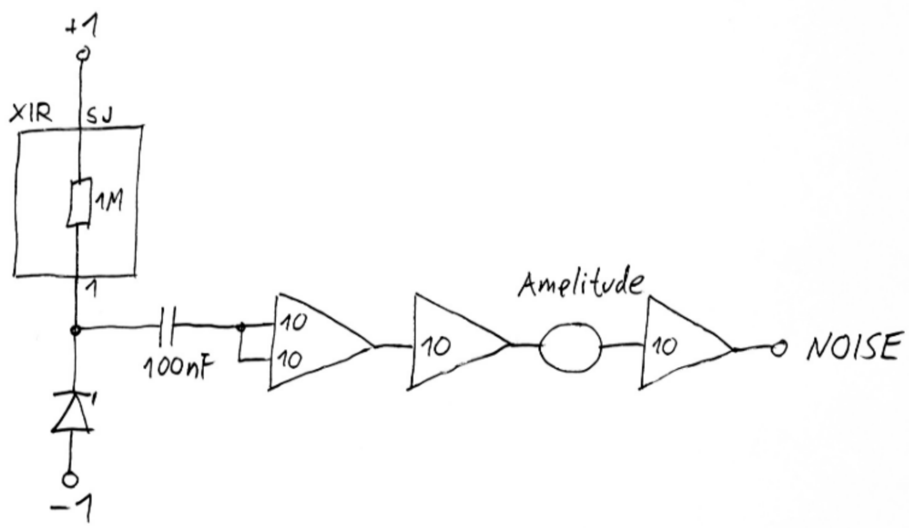
How does it work?



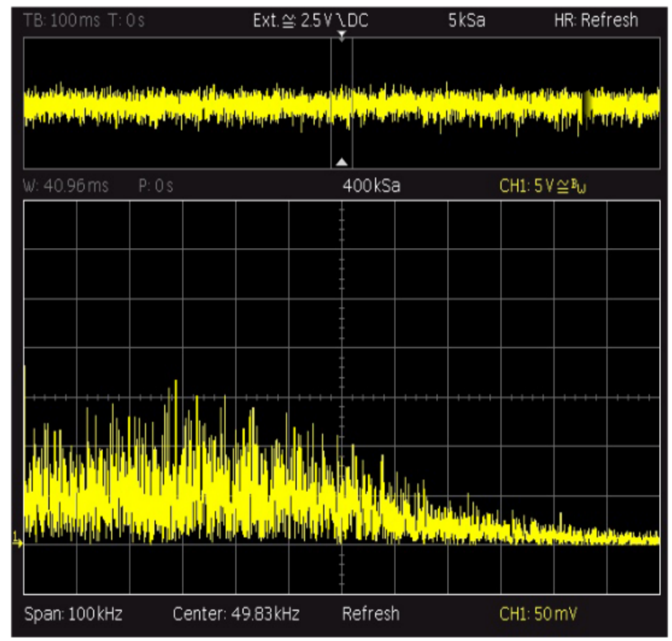
- **The variable pause is created by integrating with a constant slope (-0.3) from zero to a variable peak level, which depends on the input signal.**
- **The fixed pulse duration is created by integrating with a variable slope (depending on the input signal) from the variable peak down to zero. The combination of variable peak level and variable slope gives a fixed duration.**
- **The left comparator selects the slope for integration.**
- **The middle comparator selects the digital output level. The +1 input signal can be changed to other levels, if at the same time the -0.5 signal is changed to -0.5 * output level.**
- **The right comparator selects the compare level for determining the end of the integration slope.**

5.20 Noise Generators

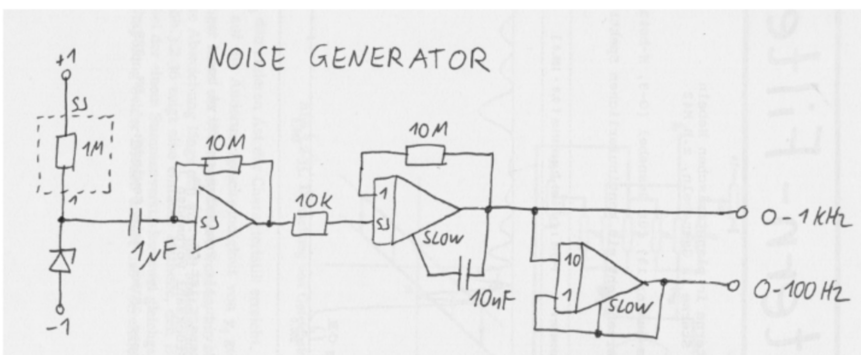
This is a noise generator. The oscilloscope screenshot shows voltage over time and the FFT spectrum. The noise signal is usable up to about 50kHz.



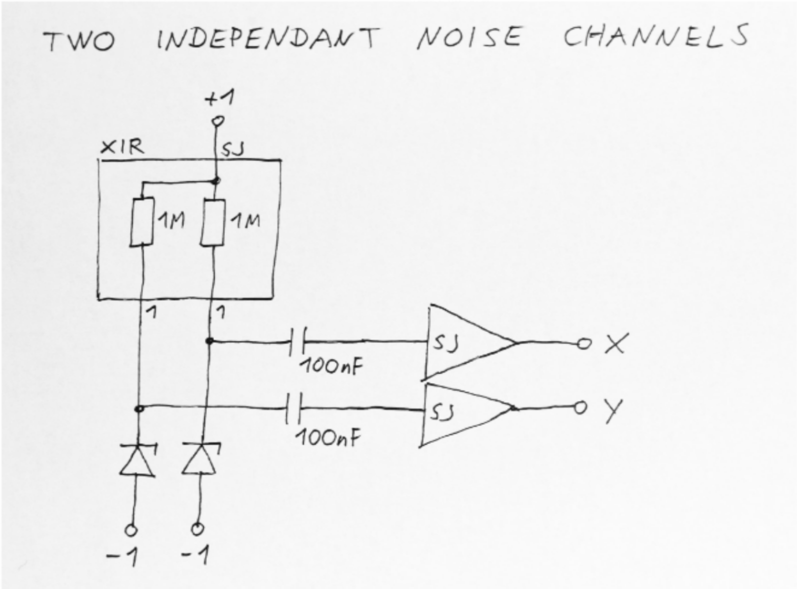
Noise

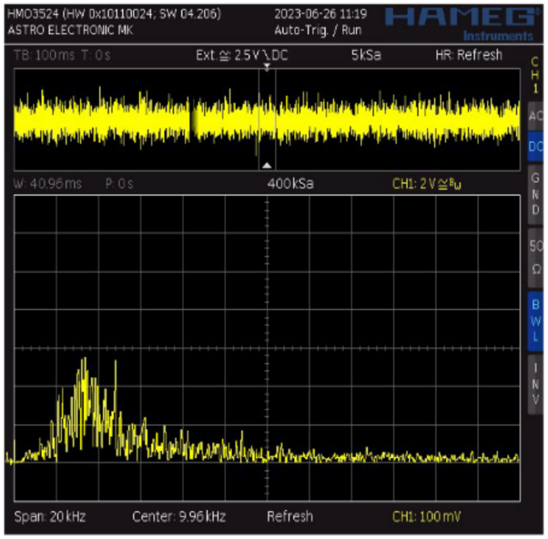


Generator with about 1kHz and 100Hz bandwidth:

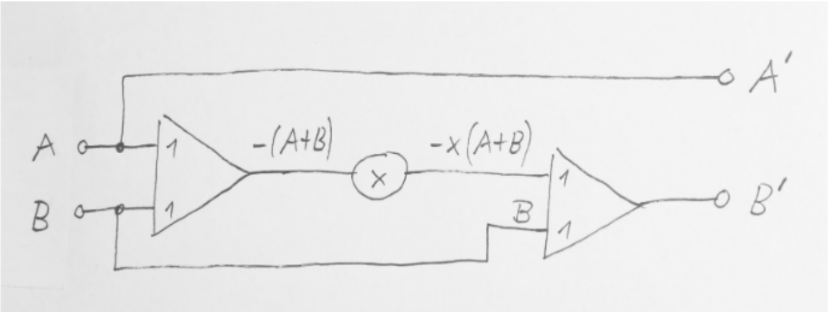


This noise generator is even simpler and has two independant channels. You need only one XIR block, two zener diodes, two capacitors and two inverters. It has a different noise spectrum with a peak at about 3kHz.



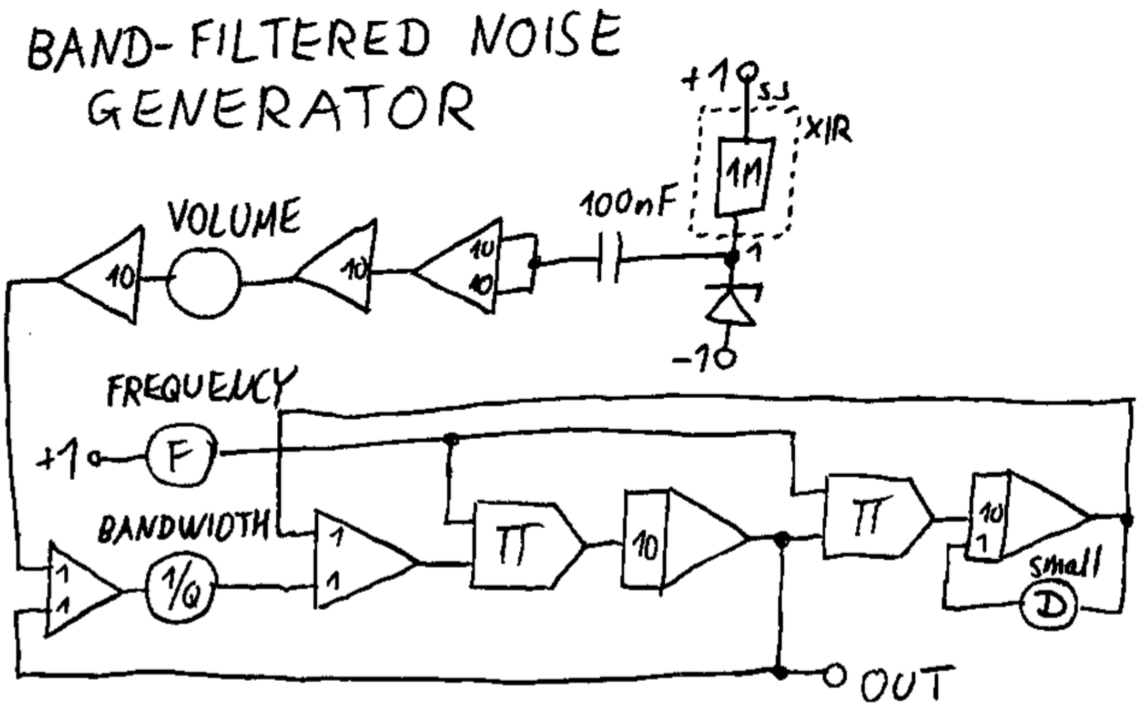


With this circuit the two independant noise signals A and B can be transformed into two signals A' and B' with adjustable degree of correlation x, from 0 to 1. If x=0, then B'=-B. If x=1, then B'=A=A'



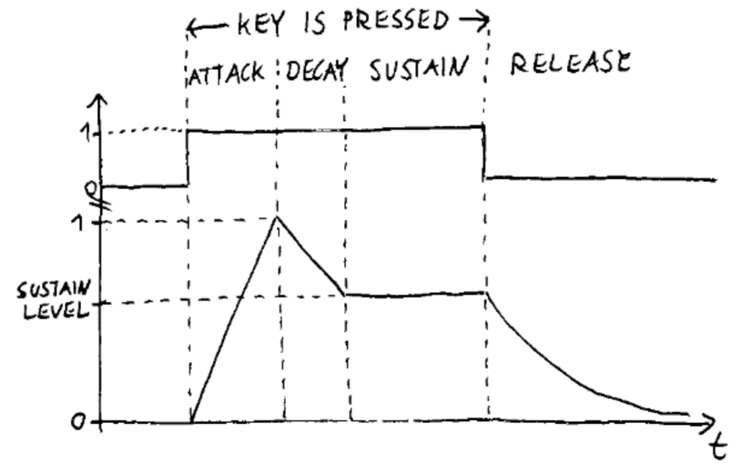
5.21 Band-Filtered Noise Generator

This is a noise generator followed by a bandpass filter with adjustable center frequency (from 0 to 1592 Hz) and quality factor (or relative bandwidth). The filter may oscillate when 1/Q is set very small. In this case you can increase the D coefficient for damping the oscillation. Normally the D coefficient should be zero.



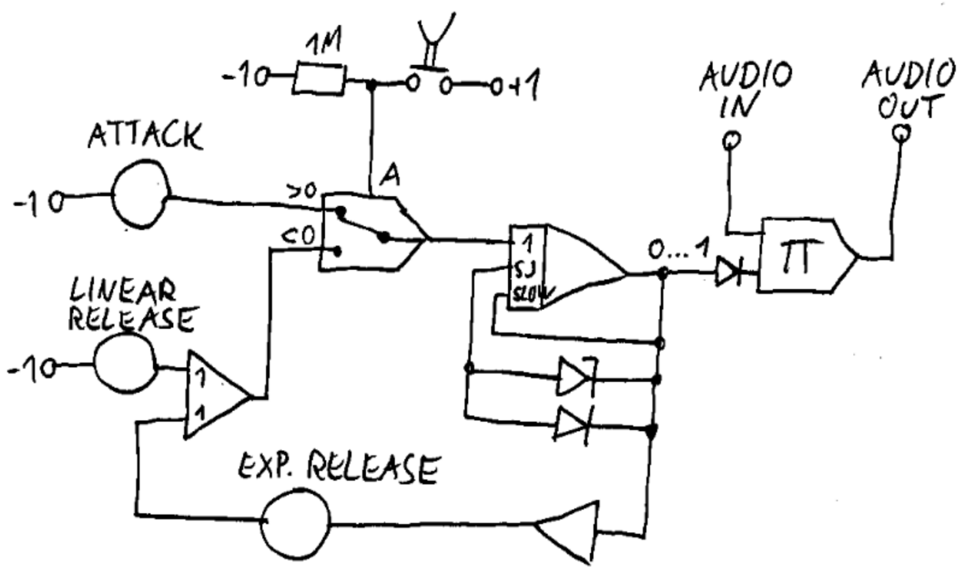
5.22 Envelope Generator for Synthesizer

The envelope consists of four segments Attach - Decay - Sustain - Release:

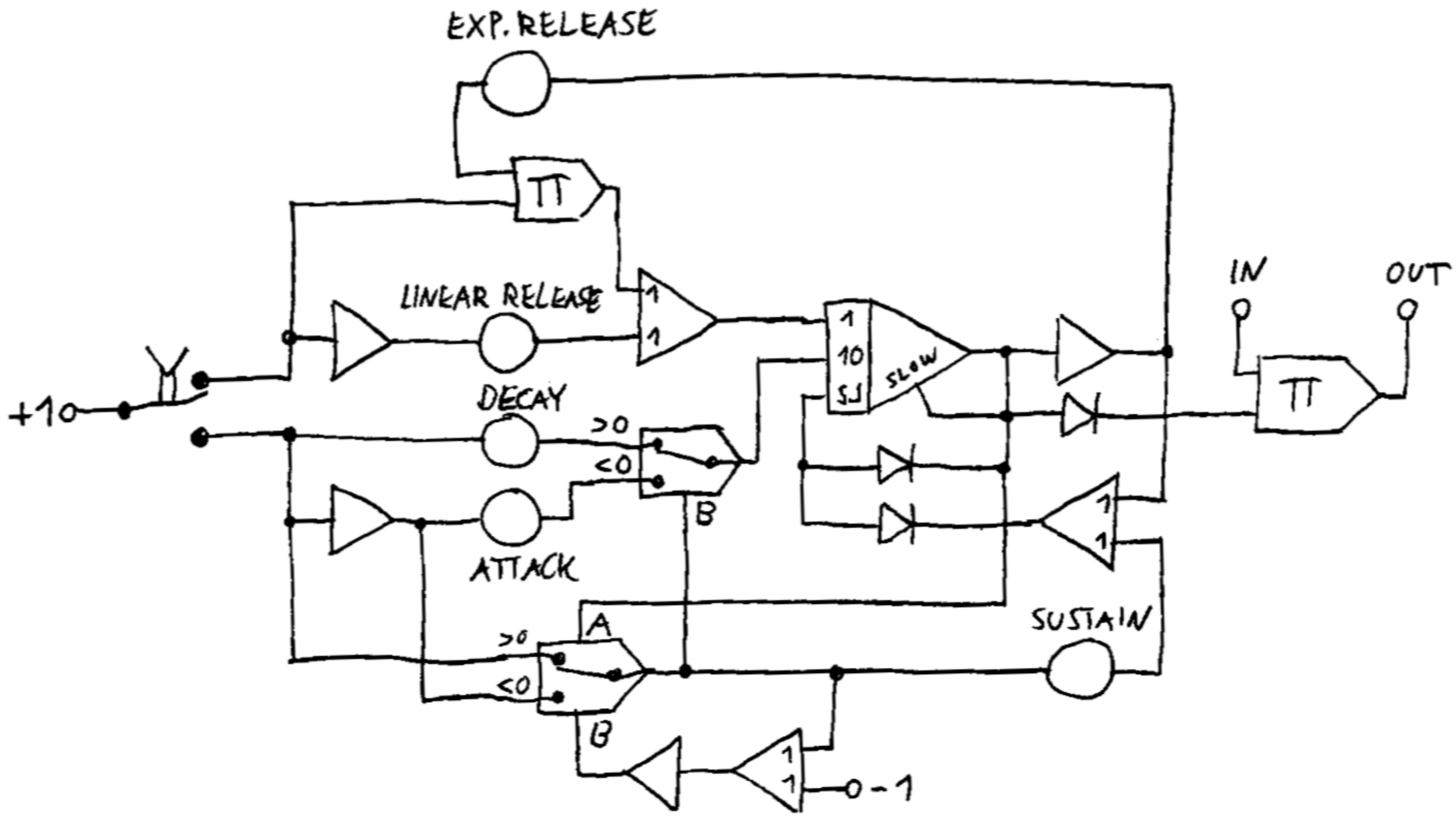


Below is the simplified version with Attack - Sustain - Release:

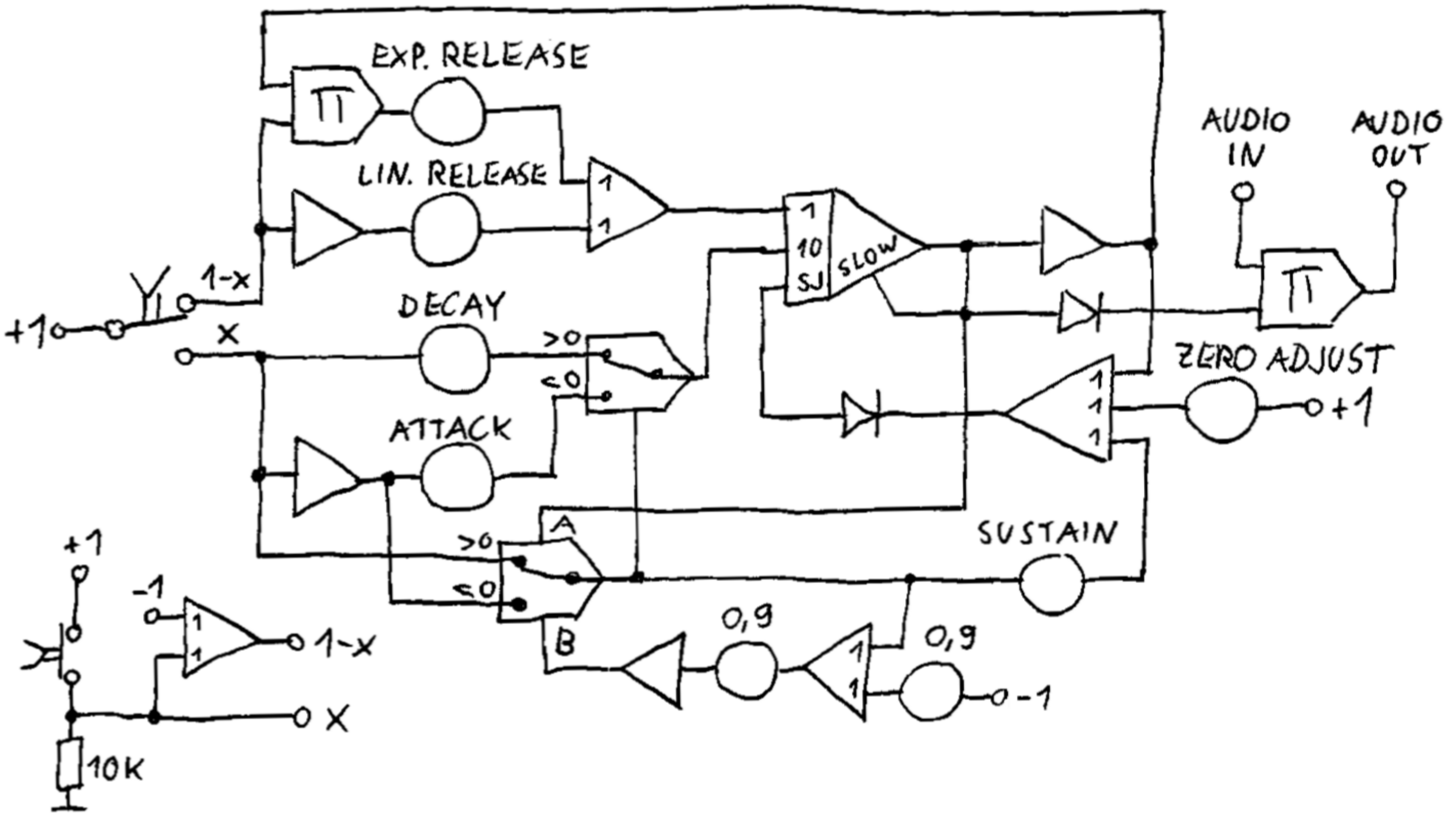
The output of the integrator can go down to about -0.7V, that's the forward voltage of the feedback diode. But we need exactly 0V at the input of the multiplier to make the output silent. That's why there is a diode at the input of the multiplier.



This is the envelope generator with four segments Attach - Decay - Sustain - Release. It doesn't work if you set the sustain level too small.



This is a slightly improved version of the previous circuit. The upper feedback diode was removed, because it was unnecessary. The lower diode is doing all the job. When the key is not pressed, the output level can be adjusted to zero with the ZERO ADJUST coefficient. It's best if you make this adjustment in exponential release mode. In the lower right corner is a circuit for a simple pushbutton input.



5.23 FFplay as Signal Generator

You can create input signals for THAT with the soundcard of a PC. The output voltage is typically about 2.7 V peak-peak. This can for example be done with a batch file which invokes FFplay. This is an example for creating a 1kHz sin/cos quadrature signal:

```
set "F=1000" :: Frequency in Hz
set "V=1" :: Volume
ffplay -f lavfi -i aevalsrc="%V%*sin(2*PI*%F%*t)|%V%*cos(2*PI*%F%*t)":c=stereo:s=48000
pause
```

For more details about FFmpeg and FFplay see also: http://www.astro-electronic.de/FFmpeg_Book.pdf

5.24 Gauss Function Generator

https://analogparadigm.com/downloads/alpaca_42.pdf

6 Scaling

6.1 Time Scaling

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 77ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: EAI Handbook of Analog Computation (from 1971), page 113ff

https://analogmuseum.org/library/eai_handbook.pdf

t = Problem time, this is the duration of the real-world problem

τ = Machine time (tau), this is the duration of the simulation on the analog computer $\tau = \beta * t$

Time scale factor: $\beta = \tau / t$

$\beta < 1$ means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems.

$\beta > 1$ means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions.

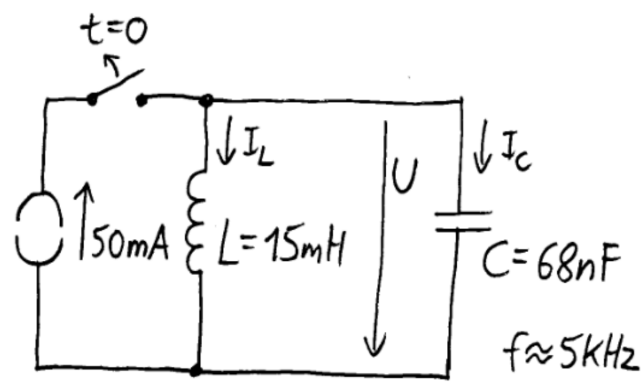
From EAI Handbook page 115:

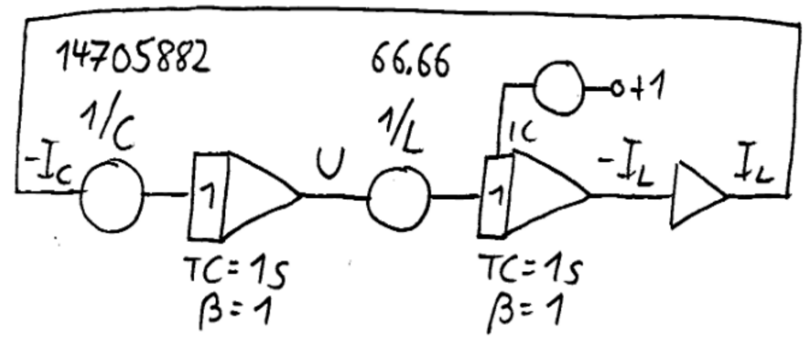
Gerneral rule: To speed up or slow down the computer solution by a factor β, every integrator input must be divided by β.

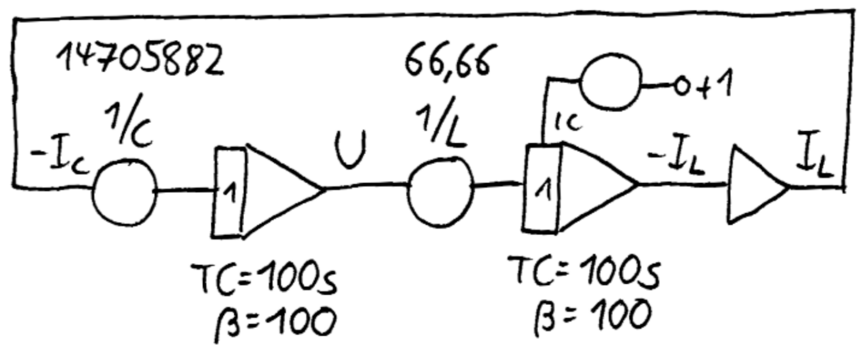
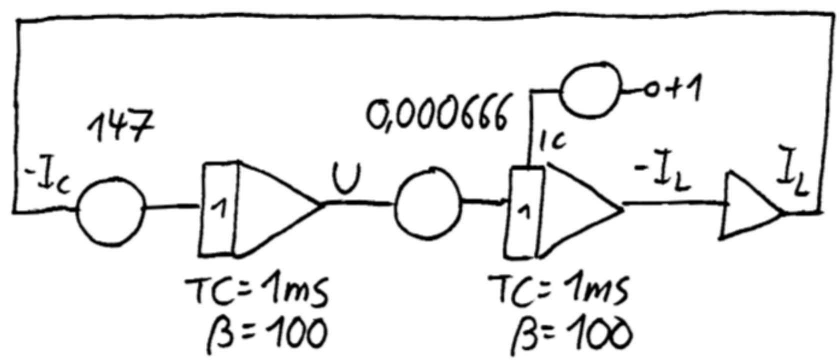
$\beta < 1$ speed up the computer solution $\beta > 1$ slow down the computer solution

Note:
There are two errors in Bernd Ulmann's book "Analog and Hybrid Computer Programming" on page 57:
In the sentence "Machine time is running faster than problem time ...", replace "$\beta > 1$" by "$\beta < 1$".
In the sentence "In other cases where the problem time is too short ...", replace "$\beta < 1$" by "$\beta > 1$".

6.2 Rules for Time Scaling, with Example

Rules	Example
Rule 1: Describe the problem with SI units. Write down the equations. If already known, write down the desired time scale factor β. β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems. β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits. If the time scale factor isn't yet known, begin with a reasonable assumption.	The current source lets 50 mA flow through the inductor. At t = 0 the switch opens. Simulate the circuit on the analog computer and find out the resonance frequency. In this example we do already know that the frequency is about 5 kHz, which is too fast for THAT. A time scale factor β = 100 seems like a good choice. Equations and given data: Inductance: $dI_L / dt = U / L$ Capacitor: $dU / dt = I_C / C$ Sum of currents: $I_L + I_C = 0$ Initial condition: I_L = 50 mA L = 15 mH C = 68 nF 

Rule 2: Draw the schematic diagram for the analog computer, under the assumption that • All integrators have a time constant TC = 1s • All integrators have a time scale factor β = 1 Write "TC = 1s, β = 1" below each integrator. The resulting schematic diagram is mathematically correct, but it won't run on the analog computer because it has many problems: • The time scale factor may be different from the desired value. • The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode). • The coefficients may be larger than +-1, or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.1 and 1. What about the integrator's time constant k0? I prefer not to use k0 because I didn't find a definition for it. It's unclear if the unit of k0 is [s] or [1/s] or if it's a dimensionless factor. If the unit of k0 is [s], then it's identical with my TC. If the unit of k0 is [1/s], then it's the inverse of my TC.	1 / (68 nF) = 14705882 F^{-1} 1 / (15 mH) = 66.66 H^{-1}  Note: The initial condition of the second integrator can be set to 0.05, but it should be connected to -1, because the integrator has an inverting output. Not +1 as in this drawing. Amplitude scaling will be described later.

Rule 3: **Change the time scale factor to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.** **If the circuit shall run in realtime (β=1), then this step is unnecessary.** **All integrators in the circuit must have the same time scale factor β.** **If the circuit contains delay lines, these must be changed as well.**	**In this example, write "TC = 100s, β = 100" below each integrator.**  **Now the time scale factor β is correct.**
Rule 4: **The time constant TC of an integrator can be multiplied by a factor X, if all inputs of this integrator are multiplied by the same factor X. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor X.** **Choose X so that the time constants of the integrators become either 1 ms or 100 ms, because these are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode).** **The integrators may have different time constants TC, but they all share the same time scale factor β.** **With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.**	**In this example the factor X is 0.00001, so that the time constants of both integrators is now 1ms, which is the default time constant of THAT's integrators.**  **Now the time constants TC of the integrators are correct, and the time scaling factor β is also correct. But there are still two problems: The first coefficient is too large, and the second coefficient is too small.**

<table>
<tr>
<td>

Rule 5:

If all input signals of an integrator (or summer) are multiplied by a factor X, then the output of the integrator (or summer) will also become larger by a factor X.
This introduces an error that must be compensated somewhere else in the circuit.

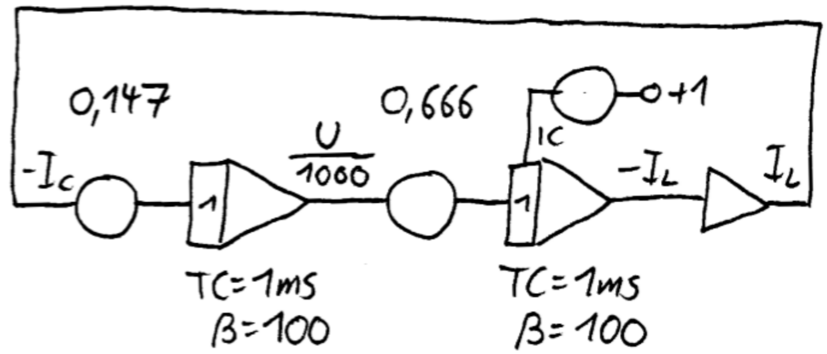
If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor X.

This operation doesn't change the integrator's time constant TC and time scale factor β.

If you can't find a solution where all coefficients are in the preferred range, you may have to choose a different time scale factor β.

</td>
<td>

In this example, multiply the value of the first coefficient by 0.001, so that the input signal of the first integrator becomes smaller, and the output is now U / 1000.
Luckily the first integrator doesn't have a non-zero initial condition, so we don't have to change it.
We compensate the error by multiplying the second coefficient by 1000, so that the input of the second integrator gets the same signal as before.



Now we can let this circuit run and measure the resonance frequency on the oscilloscope. The frequency is about 50 Hz and the period is about 20 ms.
To get the real-world frequency, we must multiply the measured frequency by the time scale factor β = 100, which gives 5 kHz.
To get the real-world period (or time), we must divide the measured period (or time) by the time scale factor β = 100, which gives 0.2 ms.

</td>
</tr>
<tr>
<td>

Rule 6:

Multiple coefficients in series can be combined into a single coefficient, which is set to the product of them.

</td>
<td>

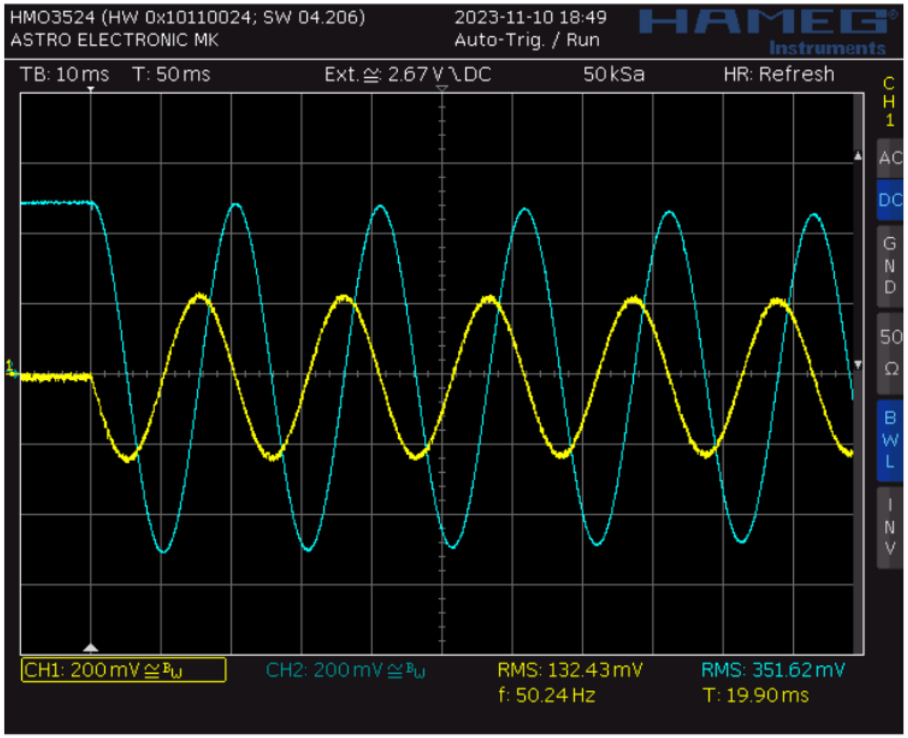
This rule doesn't apply to this example.

</td>
</tr>
</table>

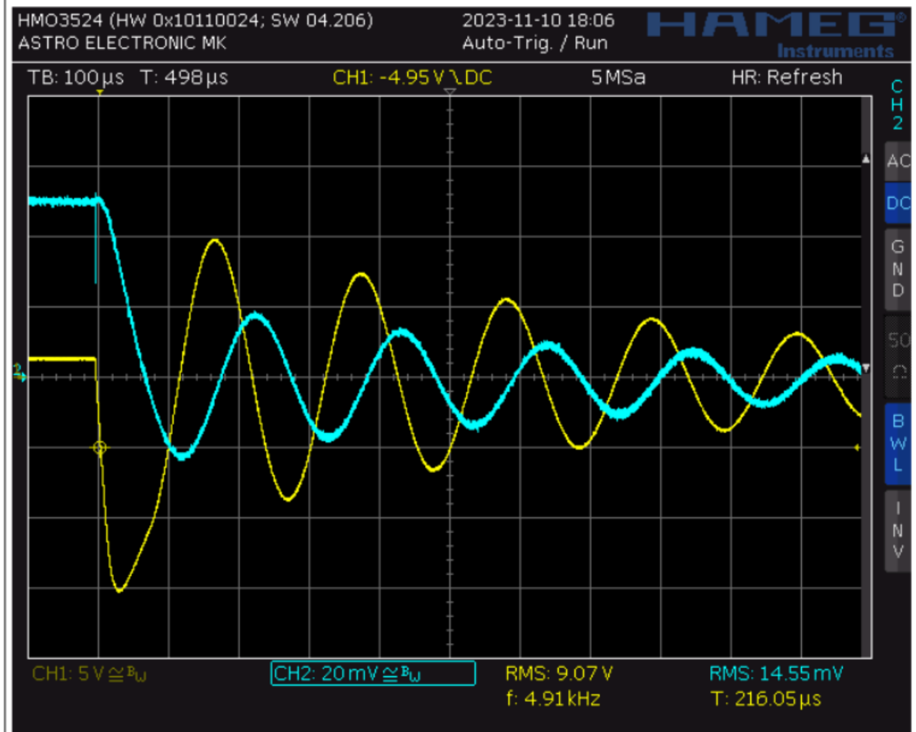
With the circuit as above (after applying rule 5) and with the initial condition set to 0.05 (Ampere) the output of the first integrator is the voltage in Millivolts (U / 1000), and the output of the second integrator is the current in Ampere.

Amplitude scaling is very simple in this example, because all amplitudes do only depend on one input. That's the initial condition of the second integrator. But 0.05 is a little bit too small for accurate results. It's better to change the amplitude scaling. For example if we change the scaling factor of the input from 1 A/mu to 0.1 A/mu, then the voltage output will change accordingly from 1 mV/mu to 0.1 mV/mu (mu = machine unit).

This is the output of the analog computer (with initial condition = -0.05).
Yellow is the output of the first integrator, which should be the voltage in Millivolts, about 1.1 div peak, which must be multiplied by 1000, which gives 1100 div peak.
Cyan is the output of the second integrator after the inverter, which should be the current in Ampere, about 2.4 div peak.
The U / I ratio is about 1100 / 2.4 = 458



This is the measurement of the real-world LC circuit. Obviously it has more damping, but that was to be expected.
Yellow is the voltage with 5 V/div.
Cyan is the current, which was measured with a 1 Ohm resistor in series with the inductor, so that 20 mV/div is equivalent to 20 mA/div.
The U / I ratio is about 450-500



6.3 Amplitude Scaling

See also: Charlesworth, A. S. (Alan Stuart): Systematic analogue computer programming, page 22ff
https://archive.org/details/systematicanalog0000char/page/n11/mode/2up

See also: EAI Handbook of Analog Computation (from 1971), page 113ff

https://analogmuseum.org/library/eai_handbook.pdf

A simple example for amplitude scaling can be found in: "A chaotic Sprott system" https://analogparadigm.com/downloads/alpaca_43.pdf

I'll describe the amplitude scaling process in more detail now:

We must distinguish between two types of variables:

- Problem variables, these are the variables from the real world. They have a range from a minimum to a maximum. They may also have a unit (m, kg, km/h...)
- Machine variables, these are scaled so that they don't exceed the working range of the analog computer, which is from -1 to +1 machine unit.

In the above paper the equations are given with problem variables x, y, z, x', y' and z':

$x' = y * z$
$y' = x - y$
$z' = 1 - x * y$

In this case it's known that the values of all variables fit in the -10 to +10 range. That means we can divide them by 10 to get the machine variables, which are in the -1 to +1 range.

It's confusing if the same names are used for problem variables and machine variables. It's better to use different names.
For example the machine variables can be indicated by a bar ($\overline{x}$) above the variable, or they can be given an index (x_M).

Problem variable in range [-10..+10]	Machine variable in range [-1..+1]	Conversion formulas	
x	$\overline{x}$	$\overline{x} = x / 10$	$x = 10 * \overline{x}$
y	$\overline{y}$	$\overline{y} = y / 10$	$y = 10 * \overline{y}$
z	$\overline{z}$	$\overline{z} = z / 10$	$z = 10 * \overline{z}$
x'	$\overline{x'}$	$\overline{x'} = x' / 10$	$x' = 10 * \overline{x'}$
y'	$\overline{y'}$	$\overline{y'} = y' / 10$	$y' = 10 * \overline{y'}$
z'	$\overline{z'}$	$\overline{z'} = z' / 10$	$z' = 10 * \overline{z'}$

Note: In this example all variables have the same scaling factor 10. In other examples different variables could have different scaling factors. For example x could have a scaling factor 10 and y could have a scaling factor 50.

Now we can substitute all variables from the given equations by their corresponding machine variables:

$x' = y * z$ ==> $10 * \overline{x'} = 10 * \overline{y} * 10 * \overline{z}$

$y' = x - y$ ==> $10 * \overline{y'} = 10 * \overline{x} - 10 * \overline{y}$

$z' = 1 - x * y$ ==> $10 * \overline{z'} = 1 - (10 * \overline{x} * 10 * \overline{y})$

Finally all three equations can be divided by 10, and we get this set of equations with machine variables:

$\overline{x'} = 10 * \overline{y} * \overline{z}$

$\overline{y'} = \overline{x} - \overline{y}$

$\overline{z'} = 0.1 - 10 * \overline{x} * \overline{y}$

When the problem has been solved on the analog computer, the results can be converted back to problem variables, using the same conversion formulas.

6.4 Gravity on other Planets - Example with Amplitude Scaling

In this example we want to simulate how fast a stone is falling from height 1m to the surface of a planet. That depends on the gravitational acceleration at the surface of the planet.

These are the formulas for the analytical solution:

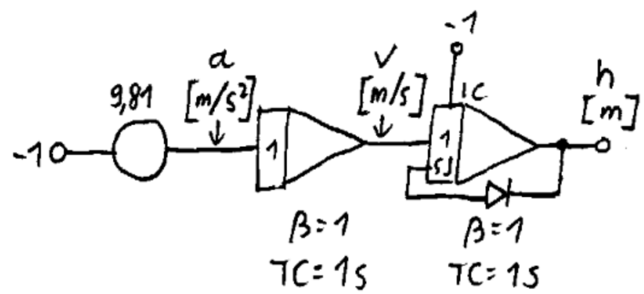
Height as a function of acceleration and time: $h = a / 2 * t^2$

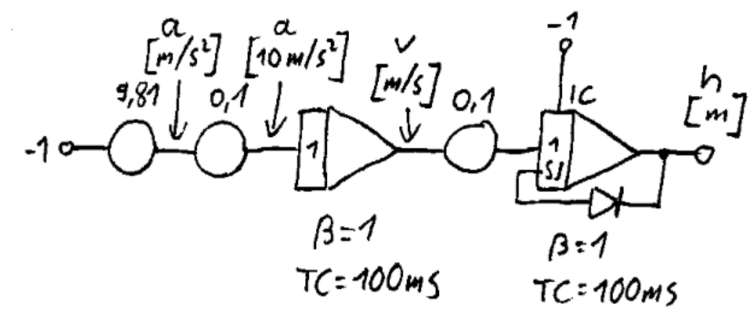
Time for falling from height h to the ground: $t = sqrt(2 * h / a)$

Velicity of falling stone after 1s: $v = a * 1s$

Impact velocity of stone at the ground, after falling from height h: $v_{max} = sqrt(2 * a * h)$

Planet	Gravitational acceleration at planet surface a [m/s^2]	Time for stone falling from 1m height [s]	Impact velocity after falling from 1m height [m/s]
Earth	-9.81	0.45	-4.43
Moon	-1.62	1.11	-1.80
Mars	-3.71	0.73	-2.72
Comet 67P/Churyumov-Gerasimenko https://en.wikipedia.org/wiki/67P/Churyumov%E2%80%93Gerasimenko Escape velocity is about 1 m/s	about 0.001	44.7	-0.045

Rules	Example
Rule 1: Describe the problem with SI units. Write down the equations. If already known, write down the desired time scale factor β. β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems. β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits. If the time scale factor isn't yet known, begin with a reasonable assumption.	At t = 0 the stone is at height h = 1m and begins to fall. The simulation ends when it reaches the ground at height h = 0. The time scale factor is β = 1 because we want to simulate in realtime. Equations and given data: Velocity: v = dh / dt Acceleration: a = dv / dt a = 9.81 m/s^2 Initial condition: h = 1m
Rule 2: Draw the schematic diagram for the analog computer, under the assumption that • All integrators have a time constant TC = 1s • All integrators have a time scale factor β = 1 Write "TC = 1s, β = 1" below each integrator. The resulting schematic diagram is mathematically correct, but it won't run on the analog computer because it has many problems: • The time scale factor may be different from the desired value. • The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode). • The coefficients may be larger than +-1, or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.1 and 1.	 Note: In this (and all following) drawings a minus sign is missing. The output signal of the velocity integrator must be -v.

Rule 3: **Change the time scale factor to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.** **If the circuit shall run in realtime (β=1), then this step is unnecessary.** **All integrators in the circuit must have the same time scale factor β.** **If the circuit contains delay lines, these must be changed as well.**	**Nothing to do in this example, because the time scale factor is already correct.**
Rule 4: **The time constant TC of an integrator can be multiplied by a factor X, if all inputs of this integrator are multiplied by the same factor X. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor X.** **Choose X so that the time constants of the integrators become either 1 ms or 100 ms, because these are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode).** **The integrators may have different time constants TC, but they all share the same time scale factor β.** **With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.**	**In this example the factor X is 0.1, so that the time constants of both integrators becomes 100ms, which is the time constant of THAT's integrators in SLOW mode.**  **Now the time constants TC of the integrators are correct, and the time scaling factor β is also correct. But there are still two problems: The first coefficient is too large, and we know that the velocity will become 4.43 m/s, which exceeds the machine unit. The velocity integrator would overflow before the stone has reached the ground.**

Rule 5:

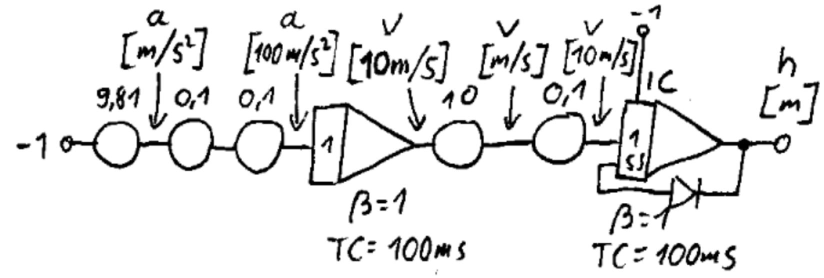
If all input signals of an integrator (or summer) are multiplied by a factor X, then the output of the integrator (or summer) will also become larger by a factor X.
This introduces an error that must be compensated somewhere else in the circuit.

If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor X.

This operation doesn't change the integrator's time constant TC and time scale factor β.

If you can't find a solution where all coefficients are in the preferred range, you may have to choose a different time scale factor β.

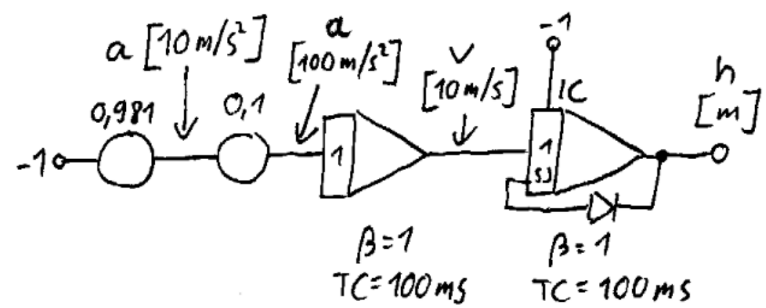
Let's solve the second problem first. We make the input of the velocity integrator smaller by a factor 10, so that its output does also become smaller by a factor 10. To compensate the error, we multiply the output signal by a factor 10.



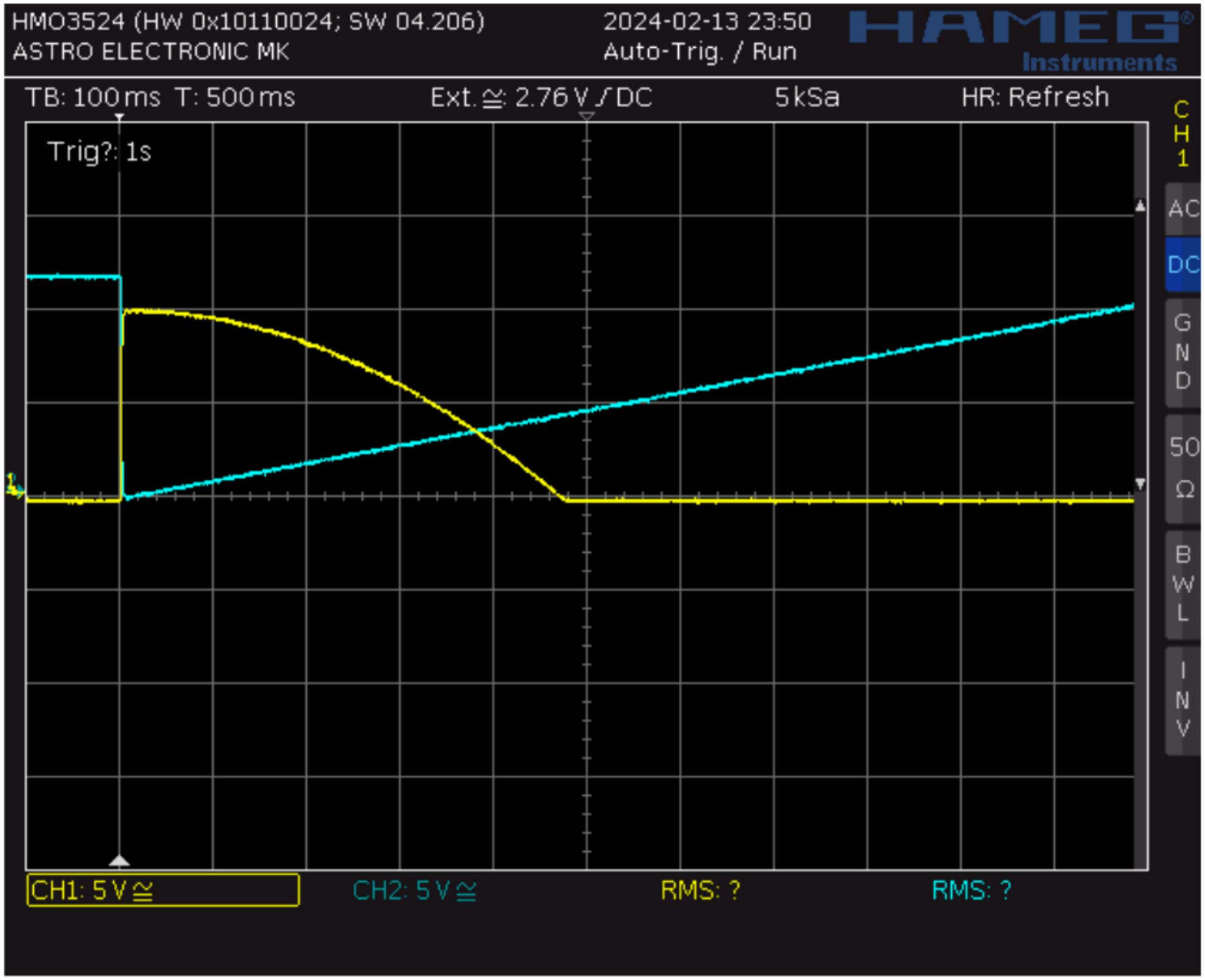
Rule 6:

Multiple coefficients in series can be combined into a single coefficient, which is set to the product of them.

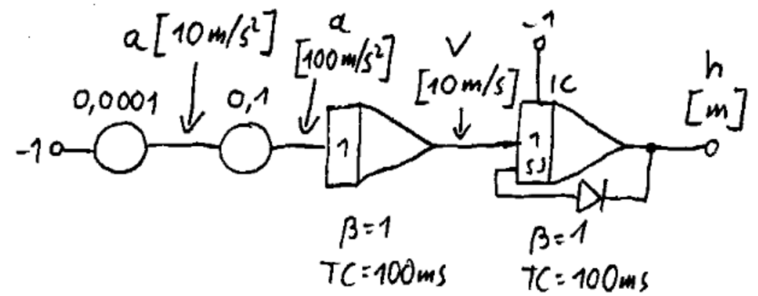
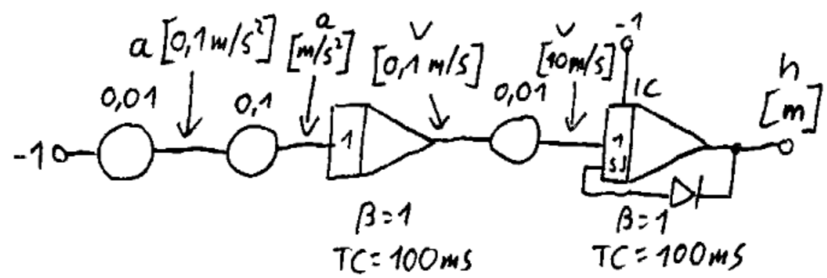
We combine the 9.81 and 0.1 coefficients to a single 0.981 coefficient, which has now a different unit [10 m/s²]. The other 0.1 coefficient remains as it is.
At the output of the velocity integrator, the 10 and 0.1 coefficients compensate each other and can be removed. The unit of the velovity signal is now [10 m/s].



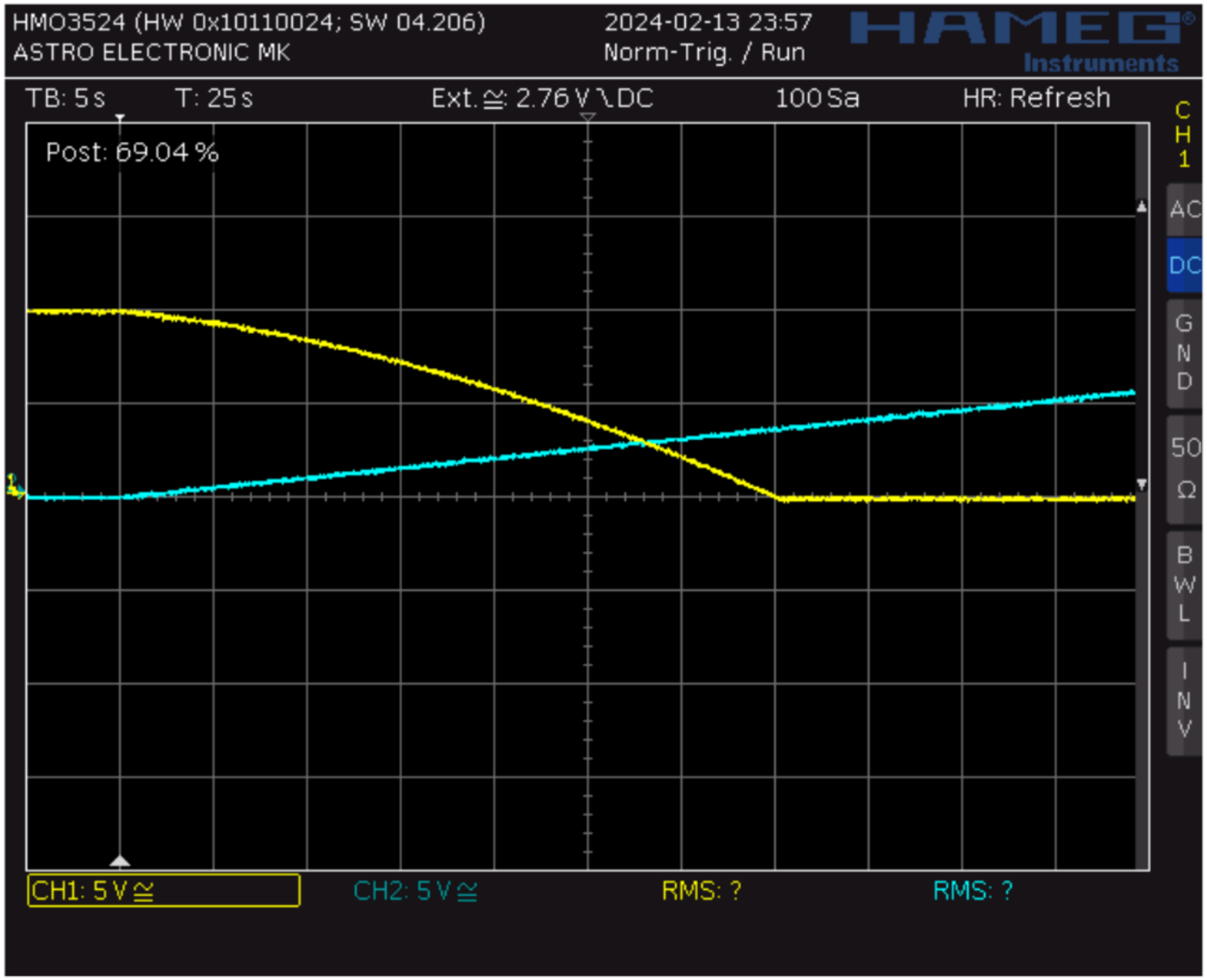
This is the result of the simulation for acceleration = 9.81 m/s². Yellow is the height and cyan is the inverted velocity. It takes about 4.7s for the stone to fall from 1m height. Which is in good agreement with 4.43s from the analytical solution.



Now let's try the same simulation for comet 67P/Churyumov–Gerasimenko, which has only 0.001 m/s² gravitational acceleration, as opposed to 9.81 m/s² on earth.

The gravitational acceleration is 0.001 m/s² . The unit of the first coefficient is 10 m/s² . That means the coefficient must be set to 0.0001, which is much too small for good computing accuracy. What can be done?	
Change the unit of the gravitational acceleration from 10 m/s² to 0.1 m/s² . Now the coefficient is 0.01, which is an acceptable value. The input of the velocity integrator is too big by a factor 100, and the output is too big by a factor 100 as well. To compensate this error, we insert a 0.01 coefficient at the output. We must check that the velocity integrator's output doesn't overflow. The expected impact velocity is -0.045 m/s, that's -0.45 machine units because the unit is 0.1 m/s.	

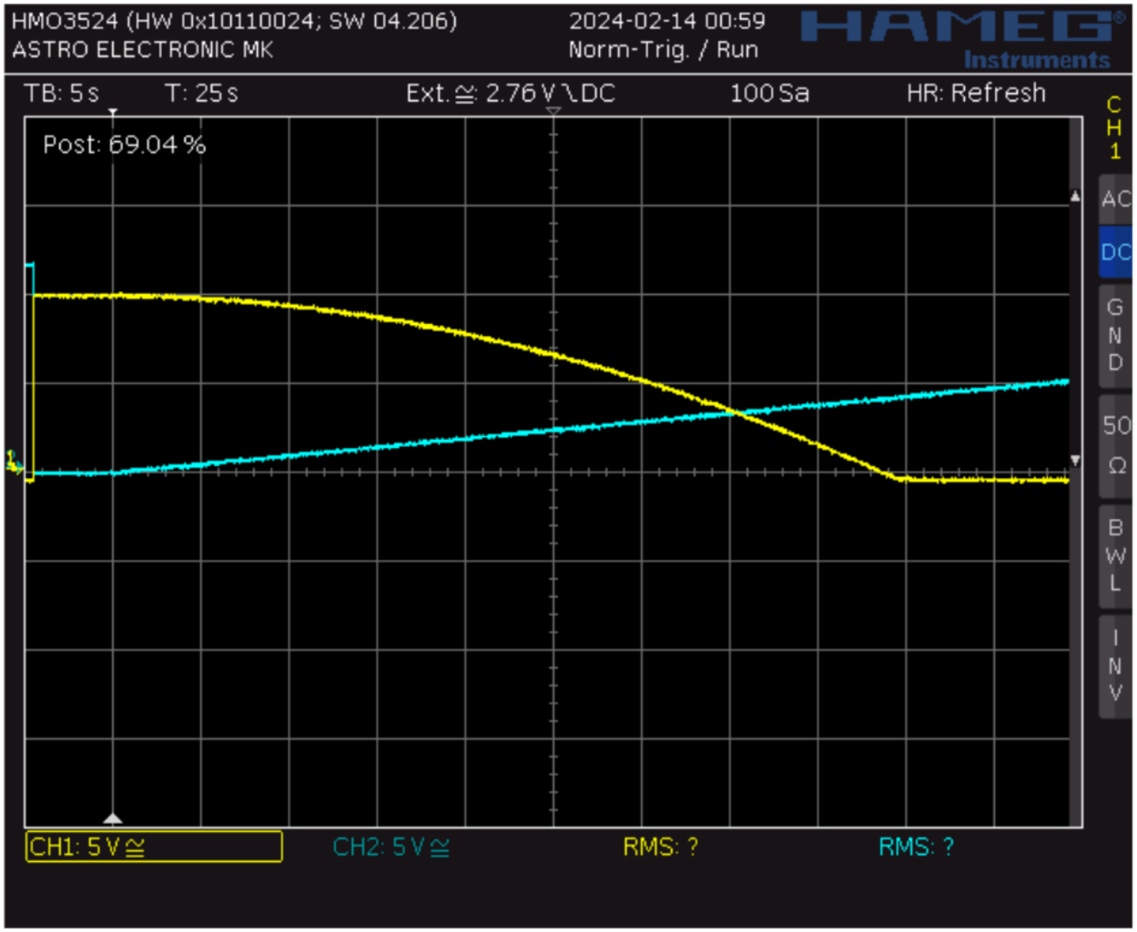
This is the output of the simulation for comet 67P/Churyumov–Gerasimenko. It takes about 35s for the stone to fall from 1m height. Which is not in good agreement with 44.7s from the analytical solution.



What's wrong? I figured out that the reverse current of the diode is the problem. The diodes in THAT are BAS170W Schottky diodes, which have a maximum reverse current of 100 nA.

In this case it's better to use a 1N4148 diode, which has a maximum reverse current of 25 nA, or even better a BAS34 diode which has an ultra-small 1 nA reverse current.

This is the output with a BAS34 diode. 47s is in good agreement with 44.7s from the analytical solution:



For educational purpose I want to demonstrate how slow the stone is falling. The stone hangs at the end of a string which is wound on a drum, rotated by a servo motor.

ASME-MXA servo motor:

The maximum speed at 24V is 0.72s per revolution, or 1.3888 revolutions per second. If a rope drum is attached to the axis, its diameter must be 229mm for a rope speed of 1 m/s.

If the travel range is 1m for 10 revolutions, the diameter of the rope drum must be 31.8mm. The total time for 10 revolutions is 7.2s.

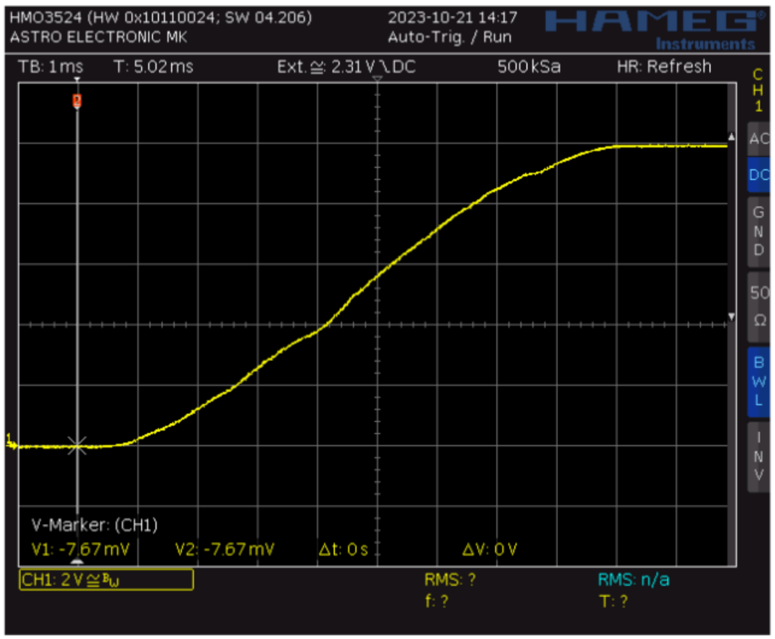
(not yet tested)

6.5 Uphill Mountainbiking - Example with Amplitude Scaling

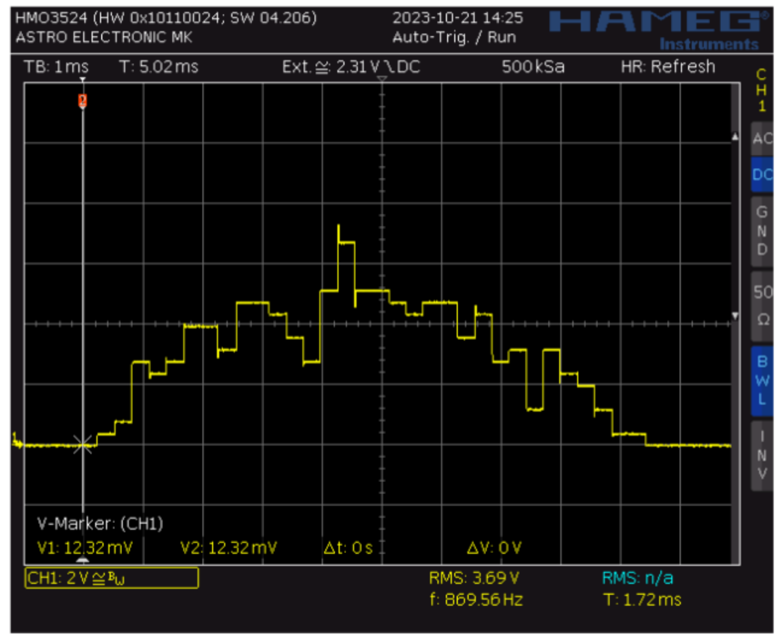
This example is a simulation for uphill mountainbiking. My favourite hill for training is the Eichelnkopf near Herzberg in the Harz Mountains in central Germany. It has 53.2 m elevation over 350 m distance, with a maximum slope of about 34%-35%. I did make three barometric measurements of the elevation profile and then calculated the average profile. This profile was programmed as a look-up-table in my Teensy LC circuit for THAT. For details see the Teensy book: http://www.astro-electronic.de/Teensy_Book.pdf

The look-up-table contains either elevation over distance, or slope over distance. In the following example I'm using the slope over distance function. The scaling factor for the distance is 1 machine unit per 350m, and the scaling factor for the slope output is 1 machine unit per 50% slope.

This is the elevation profile as a function of distance:



This is the slope as a function of distance. It's always positive because the elevation profile is monotonically increasing:



All variables must be properly scaled:

	Problem variables (in real world)	Conversion formulas		Machine variables (in analog computer)
Distance	D = 0 to 350m	D_M = D / 350m	D = D_M * 350m	D_M = 0 to 1
Slope	S = 0 to 0.5 (0 to 50%)	S_M = S * 2	S = S_M * 0.5	S_M = 0 to 1
Time Time scaling factor: β = 0.001	Problem time t = 0 to about 300s	τ = β * t	t = τ / β	Machine time τ = 0 to about 300ms
Total mass of cyclist and bicycle	M = 0 to 200kg	M_M = M / 200kg	M = M_M * 200kg	M_M = 0 to 1
Velocity	V = 0 to 2 m/s	V_M = V / (2 m/s)	V = V_M * 2 m/s	V_M = 0 to 1

Rules	Example
Rule 1: Describe the problem with SI units. Write down the equations. If already known, write down the desired time scale factor β. β < 1 means the calculation on the analog computer is faster than the real-world problem, for example when simulating population dynamics or astronomical problems. β > 1 means the calculation on the analog computer is slower than the real-world problem, for example when simulating fast chemical reactions or electronic circuits. If the time scale factor isn't yet known, begin with a reasonable assumption.	Let's assume that the cyclist (that's me) is ascending the mountain with a constant velocity. I know that it takes about 5 minutes to the peak. That means the velocity is about 350m in 300 seconds, which is a little bit more than 1 m/s. I seems to be a good choice to allow velocities up to 2 m/s in the simulation. We want to calculate the power over time. Power can be calculated with this formula: P = M * g * S * ΔD / Δt = M * g * S * V with M = total mass of cyclist and bicycle in kg (typical value is a little bit less than 100kg, but could be up to 200kg in the simulation) g = 9.81 m/s^2 S = slope as a dimensionless factor, where 1 is equivalent to 100% slope or 45° ΔD = a small horizontal distance in m S * ΔD = a small vertical distance in m Δt = a small duration in s V = velocity in m/s Let's use a time scale factor β = 0.001, so that the simulation runs 1000x faster that the real-world problem, in about 300 ms.

<table>
<tr>
<td>
<h3>Rule 2:</h3>
Draw the schematic diagram for the analog computer, under the assumption that

All integrators have a time constant TC = 1s
All integrators have a time scale factor β = 1

Write "TC = 1s, β = 1" below each integrator.
The resulting schematic diagram is mathematically correct, but it won't run on the analog computer because it has many problems:

The time scale factor may be different from the desired value.
The time constants of the integrators are different from the available time constants in THAT (1 ms, or 100 ms in SLOW mode).
The coefficients may be larger than +-1, or much too small. For good computing accuracy, the absolute values of the coefficients should be between 0.1 and 1.

</td>
<td>Note: The initial condition of the second integrator can be set to 0.05, but it should be connected to -1, because the integrator has an inverting output. Not +1 as in this drawing. Amplitude scaling will be described later.</td>
</tr>
<tr>
<td>
<h3>Rule 3:</h3>
Change the time scale factor to the desired value. To do this, multiply the time scale factors β of <u>all</u> integrators by a suitable factor X and multiply the time constants TC of <u>all</u> integrators by the same factor X.
All integrators in the circuit must have the same time scale factor β.
If the circuit contains delay lines, these must be changed as well.
</td>
<td>
In this example, write "TC = 100s, β = 100" below each integrator.
Now the time scale factor β is correct.
</td>
</tr>
</table>

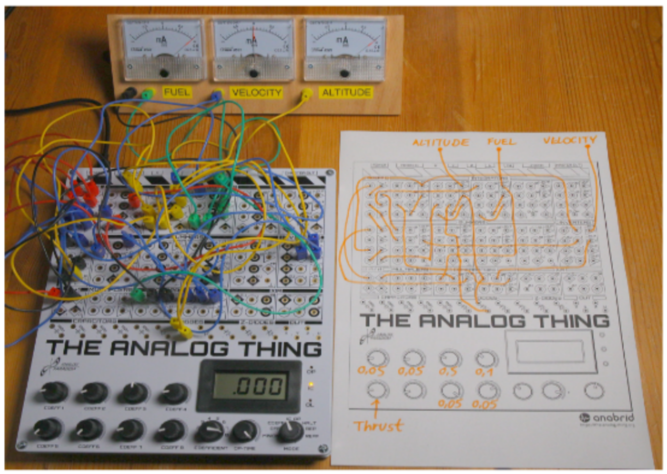
Rule 4: The time constant TC of an integrator can be multiplied by a factor X, if all inputs of this integrator are multiplied by the same factor X. This operation doesn't change the time scale factor β and the initial condition of the integrator. Each integrator can have its own factor X. Choose X so that the time constants of the integrators become either 1 ms or 100 ms, because these are available in THAT (1 ms in normal mode, and 100 ms in SLOW mode). The integrators may have different time constants TC, but they all share the same time scale factor β. With other words: The time constant TC is a property of each integrator, however the time scale factor β is a property of the whole circuit.	In this example the factor X is 0.00001, so that the time constants of both integrators is now 1ms, which is the default time constant of THAT's integrators. Now the time constants TC of the integrators are correct, and the time scaling factor β is also correct. But there are still two problems: The first coefficient is too large, and the second coefficient is too small.
Rule 5: If all input signals of an integrator (or summer) are multiplied by a factor X, then the output of the integrator (or summer) will also become larger by a factor X. This introduces an error that must be compensated somewhere else in the circuit. If the integrator's initial condition (IC) is different from 0, then it must also be multiplied by factor X. This operation doesn't change the integrator's time constant TC and time scale factor β. If you can't find a solution where all coefficients are in the preferred range, you may have to choose a different time scale factor β.	In this example, multiply the value of the first coefficient by 0.001, so that the input signal of the first integrator becomes smaller, and the output is now U / 1000. Luckily the first integrator doesn't have a non-zero initial condition, so we don't have to change it. We compensate the error by multiplying the second coefficient by 1000, so that the input of the second integrator gets the same signal as before. Now we can let this circuit run and measure the resonance frequency on the oscilloscope. The frequency is about 50 Hz and the period is about 20 ms. To get the real-world frequency, we must multiply the measured frequency by the time scale factor β = 100, which gives 5 kHz. To get the real-world period (or time), we must divide the measured period (or time) by the time scale factor β = 100, which gives 0.2 ms.

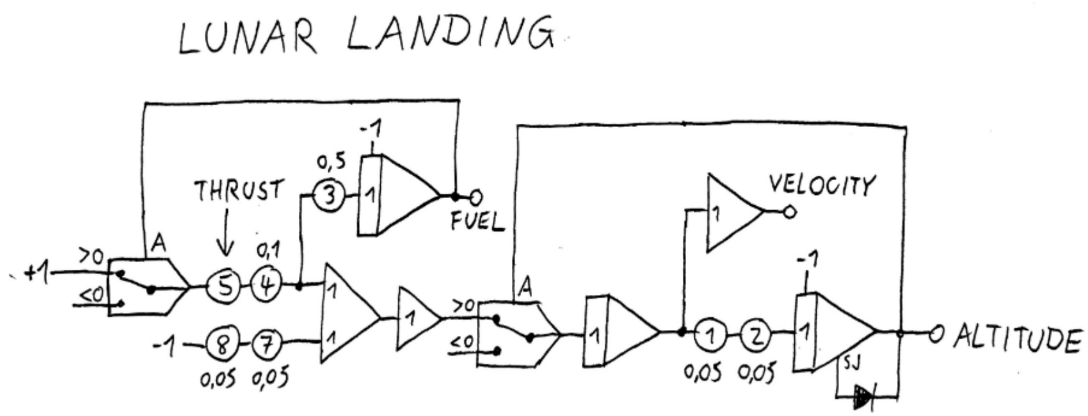
This is an unsolved problem. Amplitude scaling is very complicated. Work in progress.

7 Examples

7.1 Lunar Landing

An analog computer should have analog output devices. This is the lunar landing example from the manual, but I changed a few things. The coefficients are different, and I added a comparator at the input of the velocity integrator. If the height is negative, the input is switched to zero so that the touch-down velocity is saved. The goal of this game is to land the Eagle with the smallest possible touch-down velocity. Don't look out of the window. Look only on the instruments.



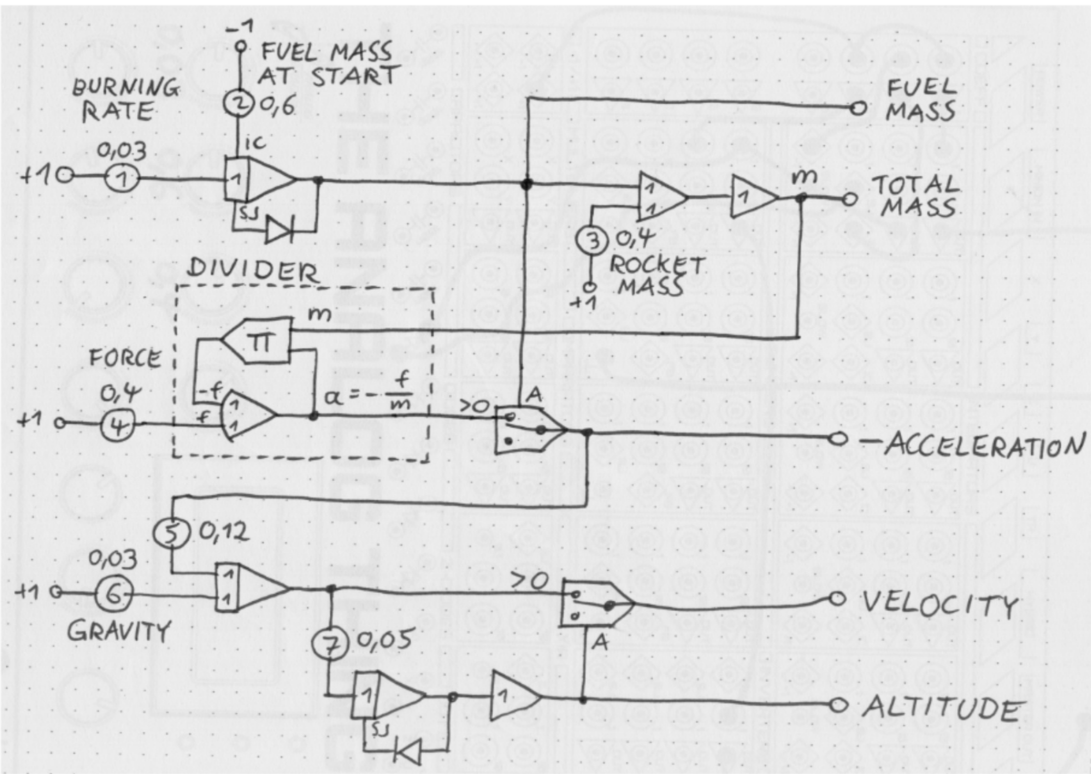


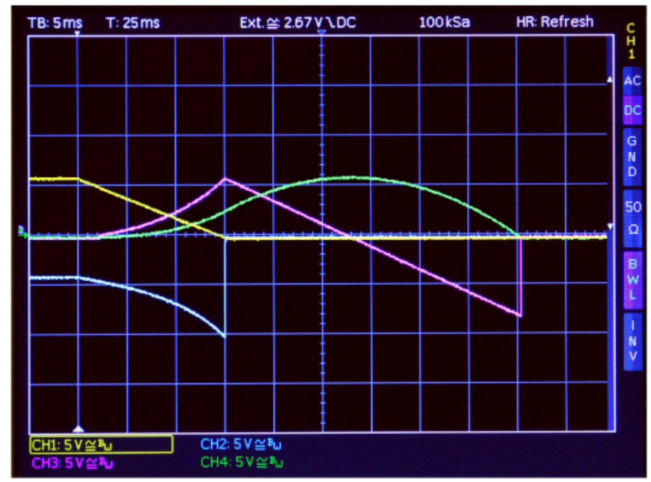
See also: https://www.youtube.com/watch?v=VNun_po0n6U

7.2 Rocket Start

This is a simulation of a rocket start, flight and crash landing. The total mass is the sum of the fuel mass (which gets less during the flight) and the mass of the empty rocket. Assuming that the motor has constant thrust, the acceleration increases because the total mass decreases. The acceleration reaches its maximum value just before the fuel tank is empty, and then drops to zero. For the rest of the flight the altitude over time is a parabola. Small error in schematic: The inverted acceleration is measured after the comparator, not before it.

Yellow = fuel mass Blue = acceleration (inverted) Red = vertical velocity Green = altitude

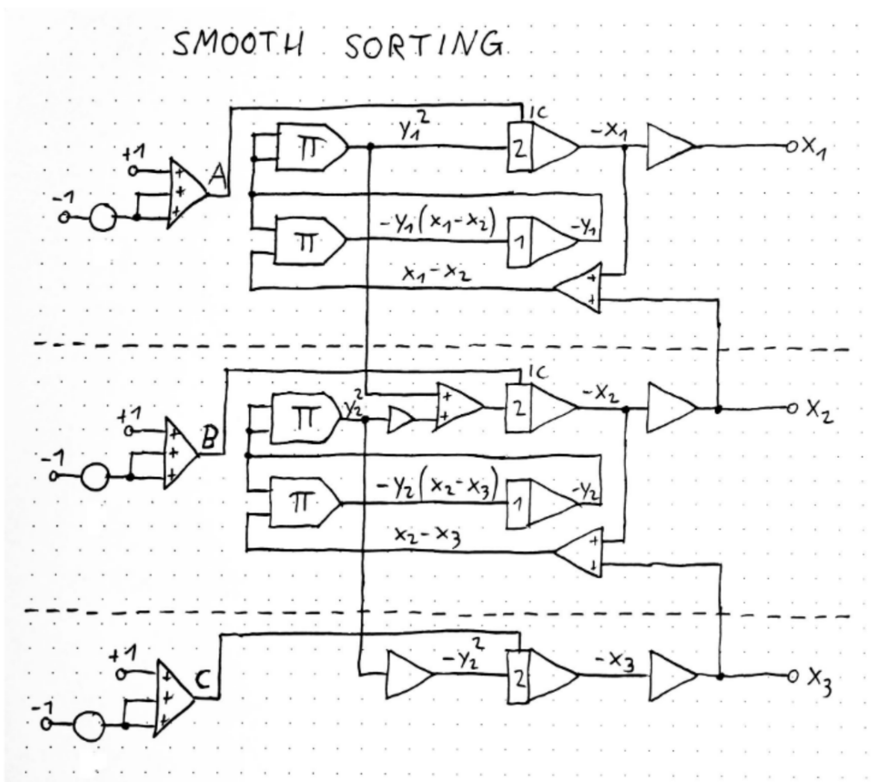


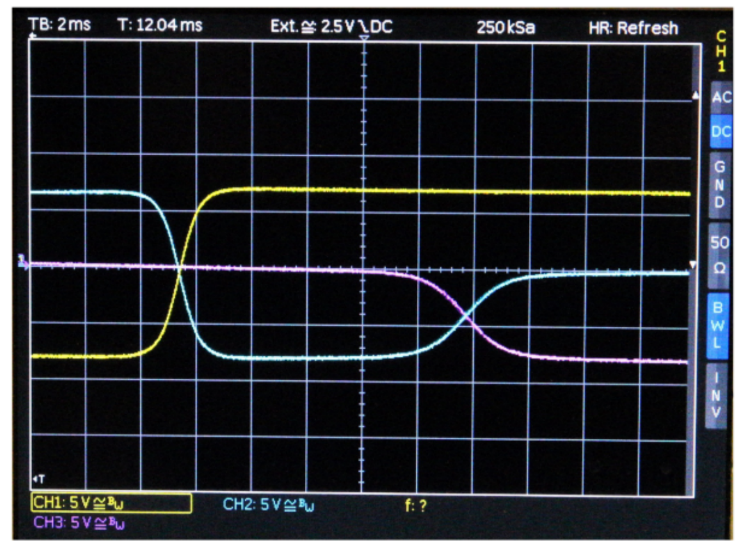


7.3 Smooth Sorting

This is an example for smooth sorting. Three input values A,B,C can be set to any values, and after some computing time will appear in sorted order at the three outputs. X1 is max(A,B,C) and X3 is min(A,B,C). This example with three inputs and three outputs runs on two THATs. If you need more inputs and outputs, you can duplicate the middle part (between the dashed lines). You can also remove the middle part, then you have only two inputs and two outputs and it will fit on one THAT.

The equations are from this paper: http://www.hrl.harvard.edu/analog/

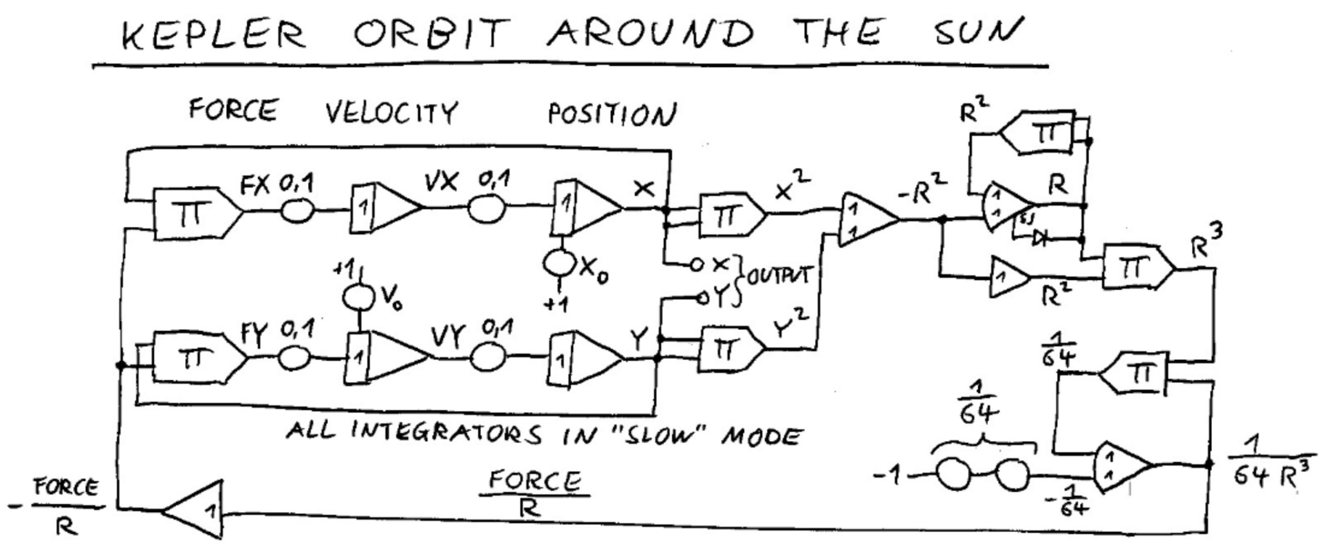




7.4 Kepler Orbits

This is a simulation of Kepler orbits (circle, ellipse, parabola, hyperbola). Unfortunately 7 multipliers are required, which means 4 THATs. I would really like to save at least one of the multipliers, but I have no idea how that could be done. (Update: Using the Teensy LC board with a look-up-table with $-R^2$ as input and $1/(64\ R^3)$ as output will save three multipliers, so that it will run on two THATs).
The sun is in the center at x=0, y=0. It's important that the distance R between the planet and the sun is always in the 0.25 to 1 range. If it's larger, then R does overflow, and if it's smaller then 1/(64*R^3) does overflow.

See also: https://analogparadigm.com/downloads/alpaca_11.pdf



R	R^2	R^3	$1/R^3$
0,25	0,062	0,0156	64
0,5	0,25	0,125	8
0,75	0,56	0,42	2,37
1	1	1	1

$0,25 < R < 1$

7.5 Gravitational Wave Simulation

This is a simulated gravitational wave from the last 10 seconds before a black hole or neutron star merger. Turn on the audio (sorry the audio is distorted because it was too loud). The left THAT solves the differential equation for frequency over time, and the right THAT is a voltage-controlled sine generator. Two THATs are required because of three multipliers. On the oscilloscope the yellow channel is frequency over time, blue is the sine wave, green is trigger signal.

Video: https://www.facebook.com/100002490928195/videos/pcb.346321967590599/447337880444874

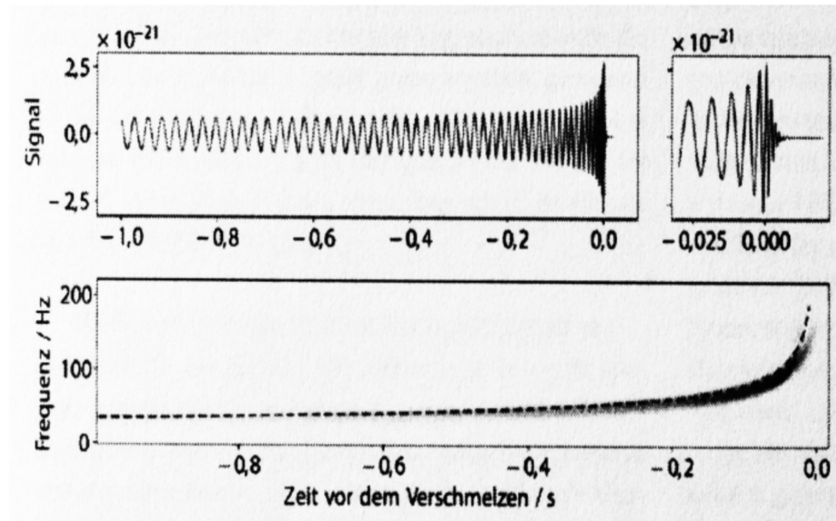
The frequency is exact, except for the last 0.1 seconds before merger. You find the formula in this paper:
https://onlinelibrary.wiley.com/.../10.1002/piuz.202101618

Die Umlaufsfrequenz nimmt wie oben beschrieben immer schneller zu und führt zu einem für das System charakteristischen Verlauf der Gravitationswelle: dem „Chirp". In grober Näherung kann dieser beschrieben werden als

$$f(t)=\frac{1}{8\pi}\cdot\left(\frac{c^3}{GM_c}\right)^{5/8}\cdot\left(\frac{5}{\delta t}\right)^{3/8},$$

wobei f die Frequenz des Gravitationswellensignals ist, c ist die Lichtgeschwindigkeit, G die Gravitationskonstante, δt die Zeit bis zur Verschmelzung. M_c ist die sogenannte Chirpmasse: eine für das System charakteristische Masse, die aus den Massen m_1 und m_2 der beiden umeinander kreisenden Objekte errechnet werden kann:

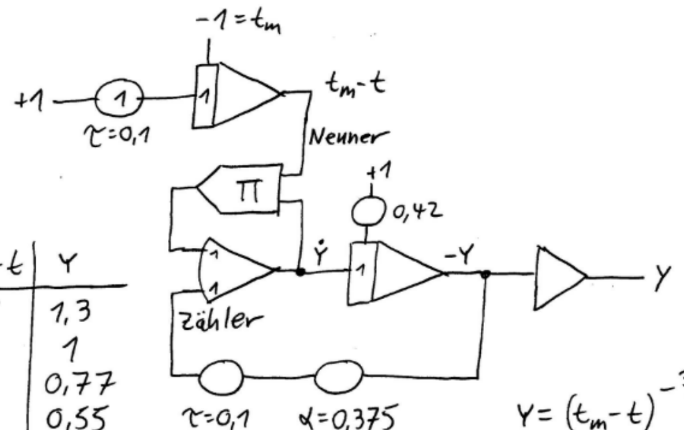
$$M_c=\frac{(m_1\cdot m_2)^{3/5}}{(m_1+m_2)^{1/5}}.$$



Quelle: https://onlinelibrary.wiley.com/doi/epdf/10.1002/piuz.202101618

The left part (calculating frequency over time) is described here (in the comments):
https://www.facebook.com/groups/305923044963825/posts/343834601172669/

The right part (voltage controlled sine generator) is described here (also in the comments):
https://www.facebook.com/groups/305923044963825/posts/341306511425478/



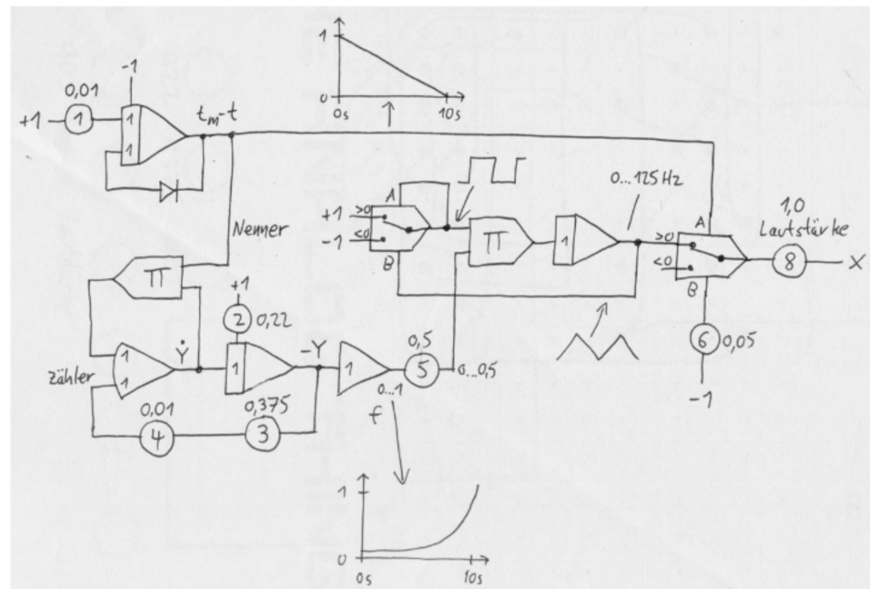
$t_m - t$	Y
0,5	1,3
1	1
2	0,77
5	0,55
10	0,42

Rechenzeit: 10ms

$$Y = (t_m - t)^{-3/8}$$

$$\dot{Y} = \frac{3}{8}(t_m - t)^{-11/8}$$

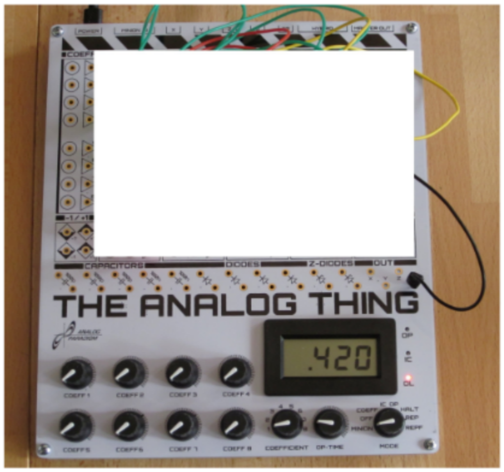
$$\dot{Y} = \frac{3}{8}\frac{Y}{t_m - t}$$

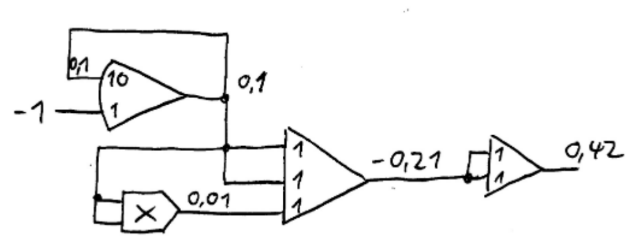


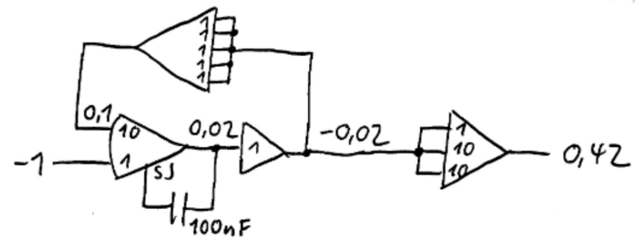
Note: If you replace the sine generator by a triangle wave generator, then only one THAT is required (see the right schematic diagram).

7.6 Calculate 0.42

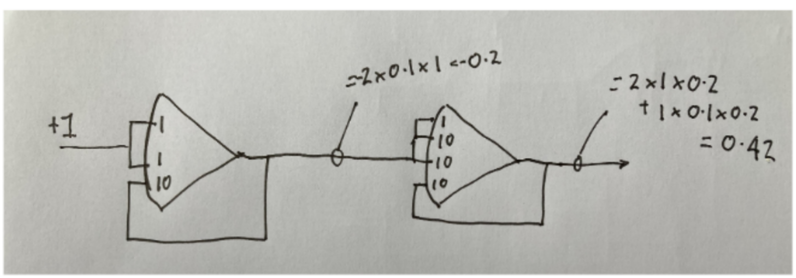
Design a circuit that calculates the answer to everything (42) normalized to 100. You are not allowed to use any coefficient potentiometers or external circuits. It must run on one THAT. Of course, there may be more than one correct solution.







This solution is from Chris Giles:

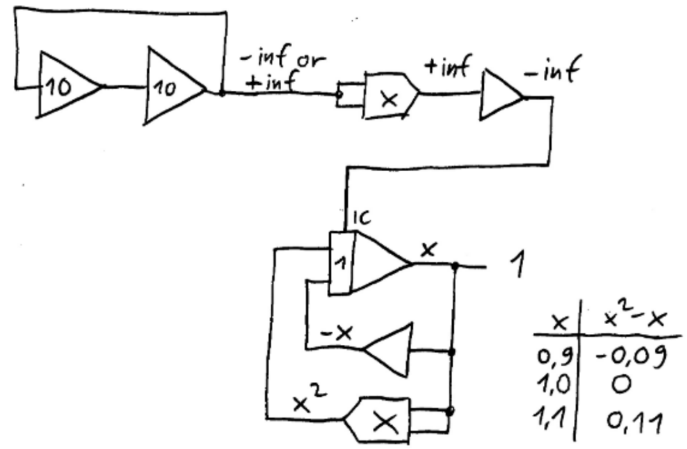


7.7 Calculate the machine unit 1

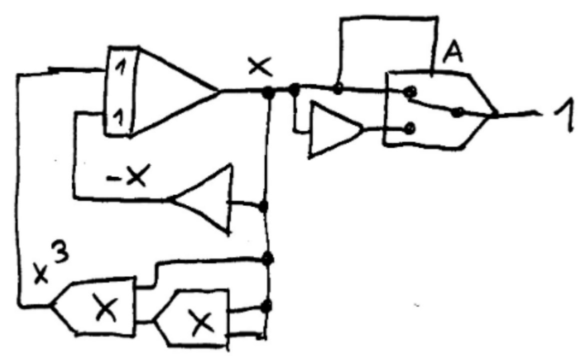
Design a circuit that calculates the +1 machine unit, without using the -1 or +1 signals. You are not allowed to use coefficient potentiometers or external components. It must run on one THAT. Small rounding errors in the last digit are no problem.

This is one possible solution. Let's first look at the lower half of the circuit. It's an integrator with x^2 - x in the feedback loop. Unfortunately this circuit has two stable results: -infinity or +1. To which state it runs depends on the sign of the initial condition. With the default zero initial condition, it may run either to -infinity or to +1. That means we must connect a negative voltage to the IC input. But it's not allowed to use the -1 signal. How can a negative signal be created from nothing? Two summers are connected in a loop with gain 100. It's almost impossible that the output voltage stays at 0 for more than a millisecond. After a very short time the output must be either -infinity or +infinity. We don't know which one, because that depends on the offset errors of the Op-Amps. But after squaring the signal, we know that the result must be +infinity, and after inverting we have -infinity.

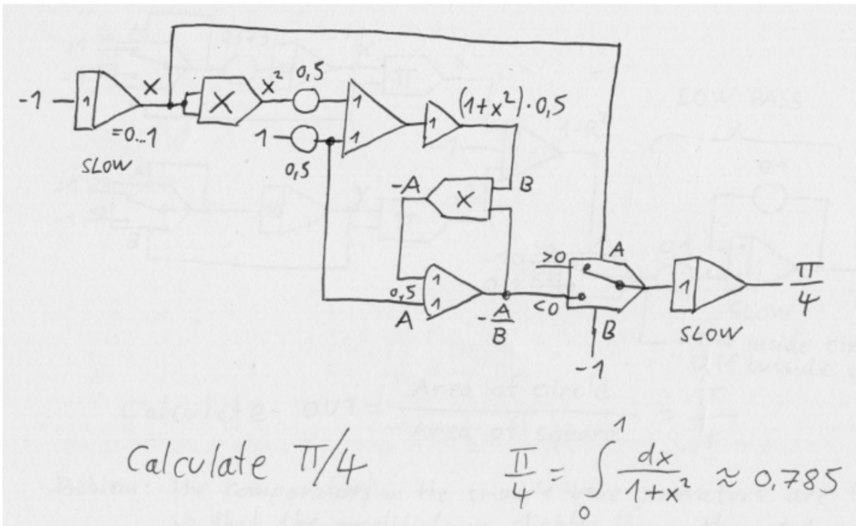
By the way, the equation x^2 - x = 0 does also have the solution x=0, but x=0 is instable in this circuit.

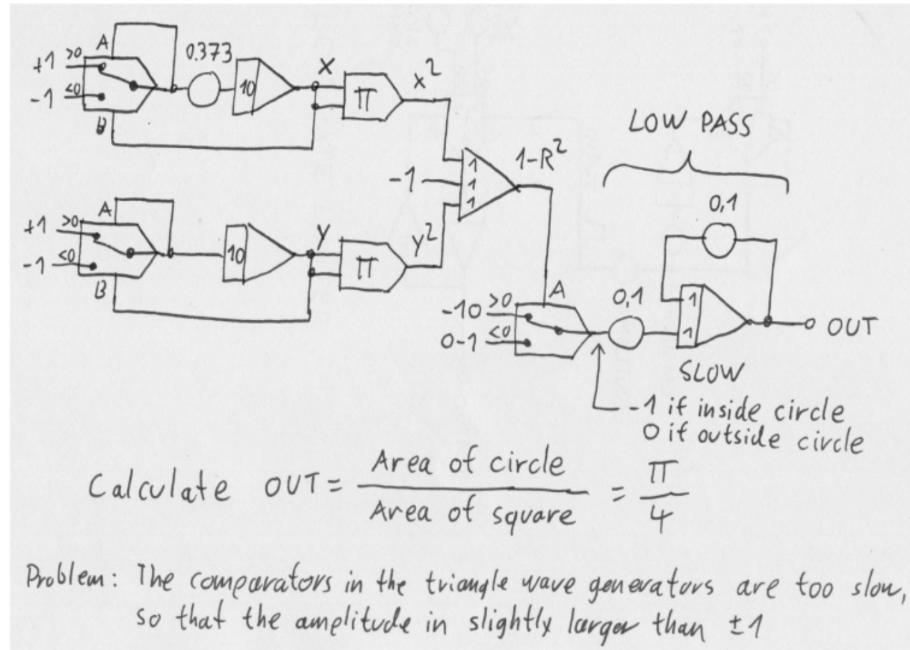


This solution is better, because it doesn't need an initial condition. The feedback of the integrator is x^3 - x, which means the integrator is running to -1 or +1. A comparator inverts the output, if negative.

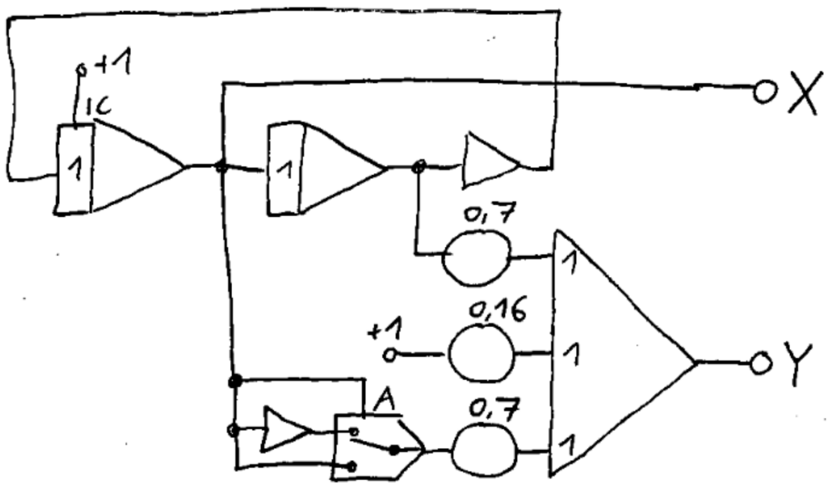


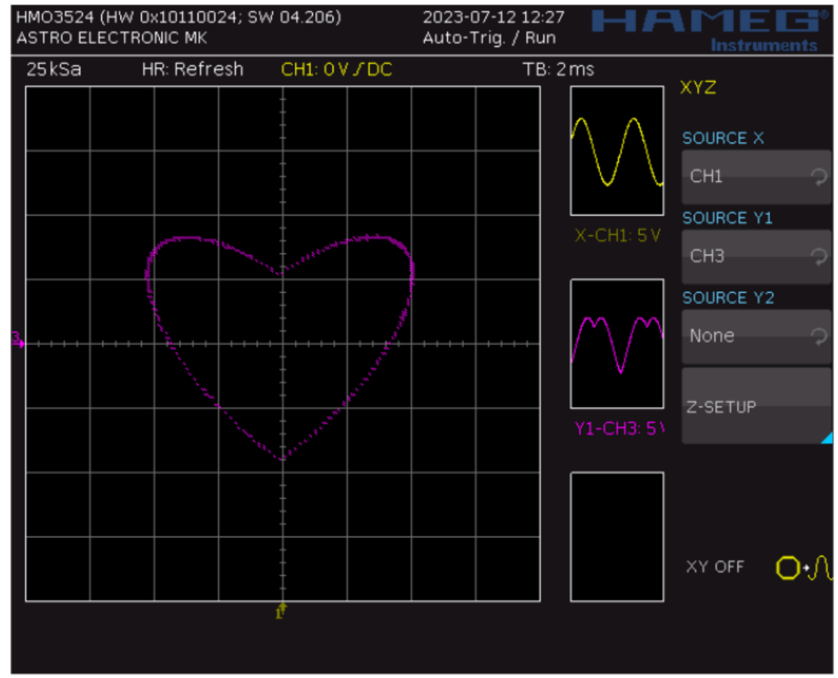
7.8 Calculate π/4



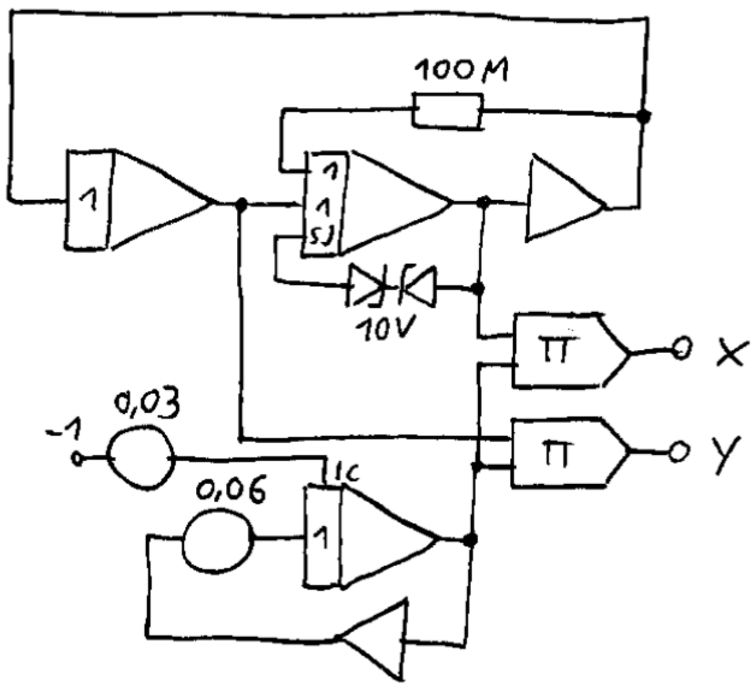


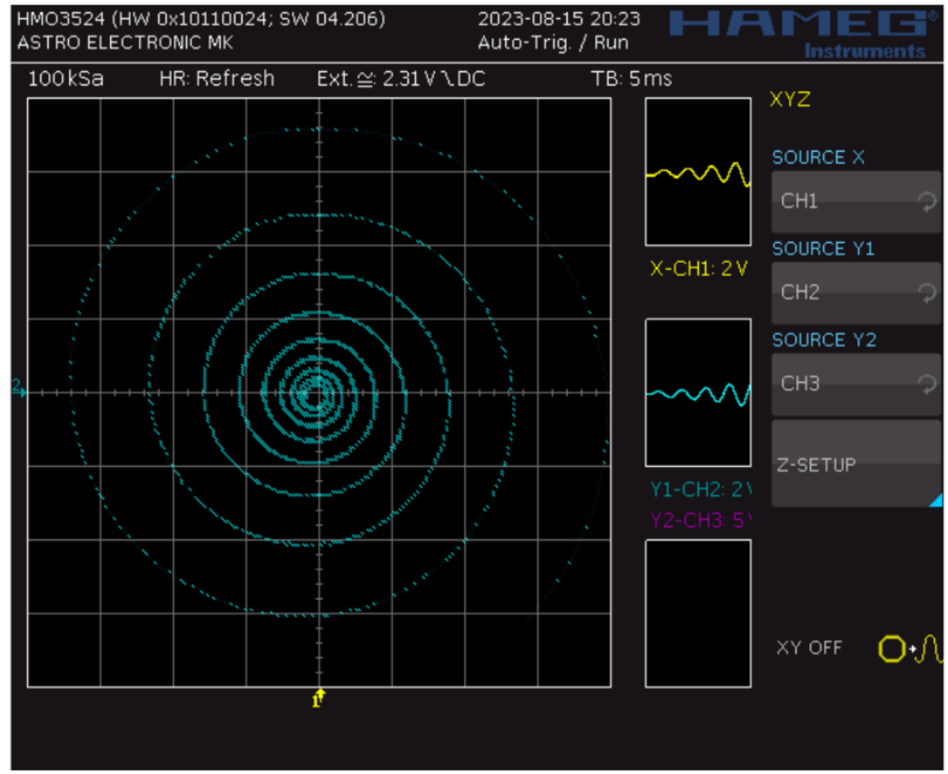
7.9 Heart Curve





7.10 Logarithmic Spiral

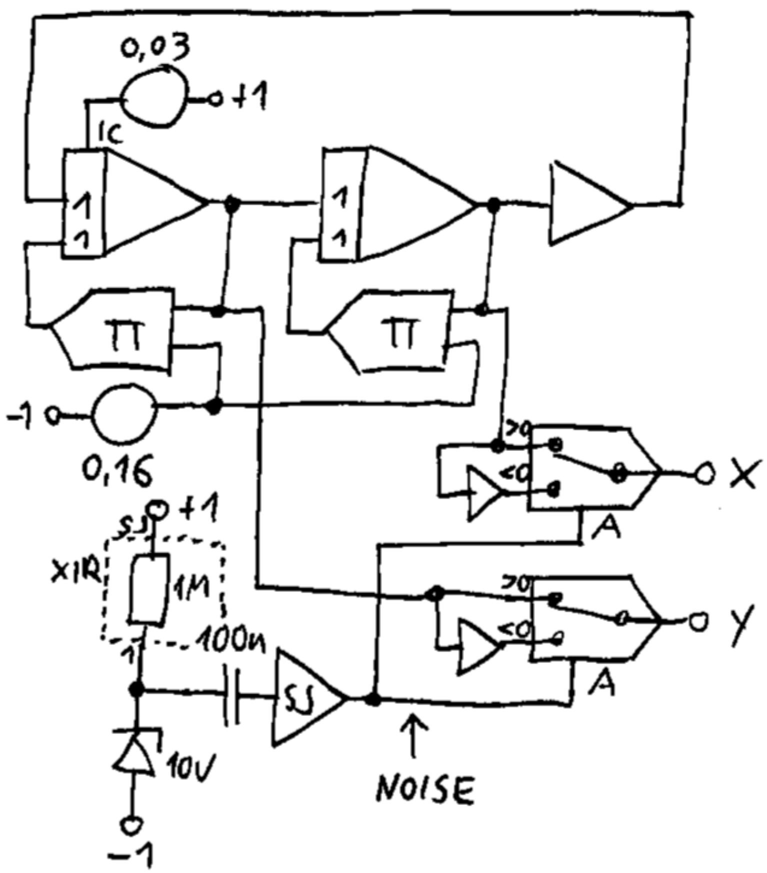


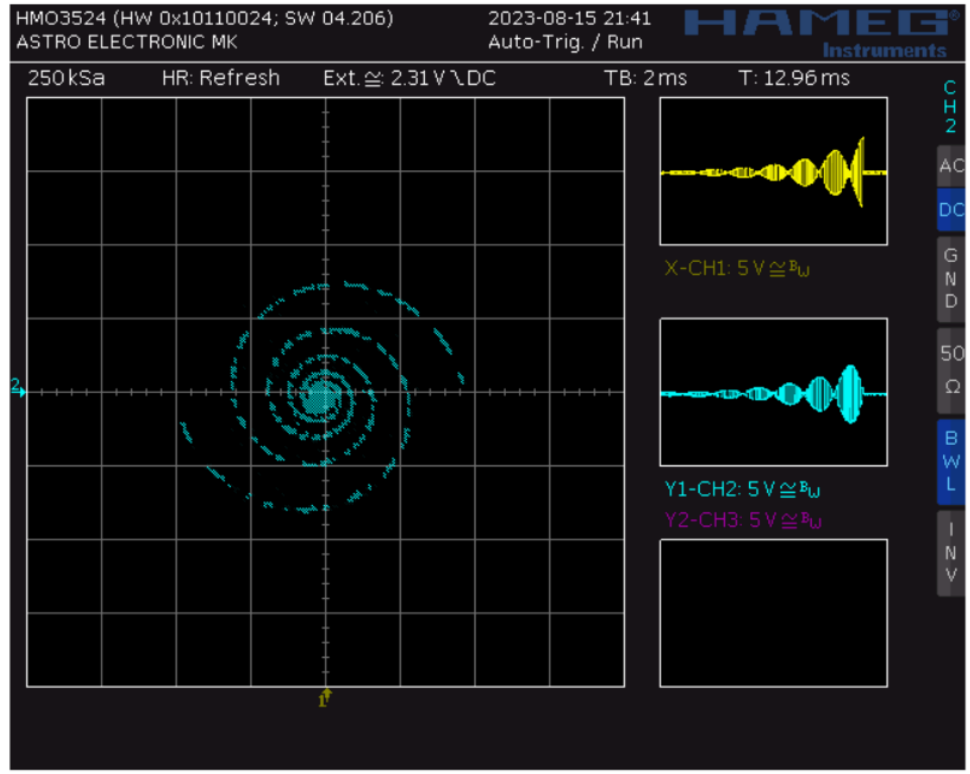


See also: https://en.wikipedia.org/wiki/Logarithmic_spiral

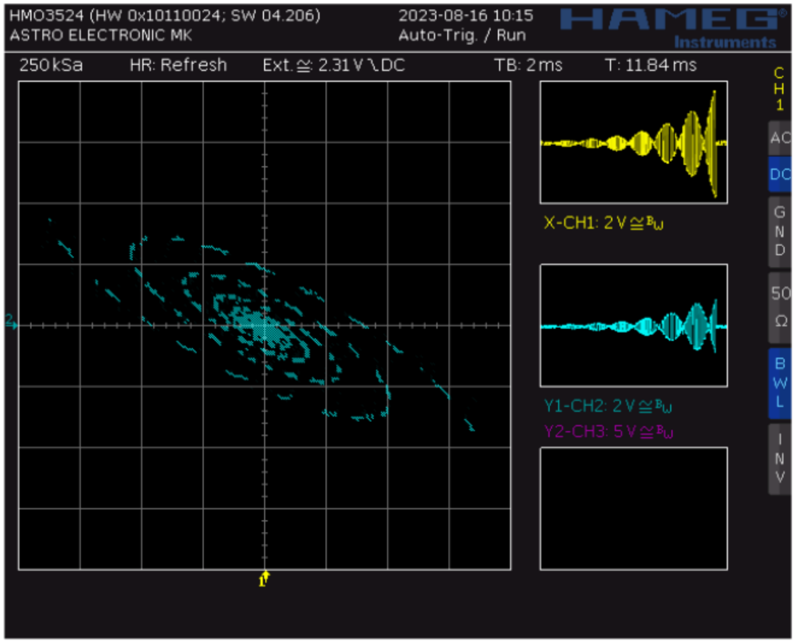
7.11 Spiral Galaxy Generator

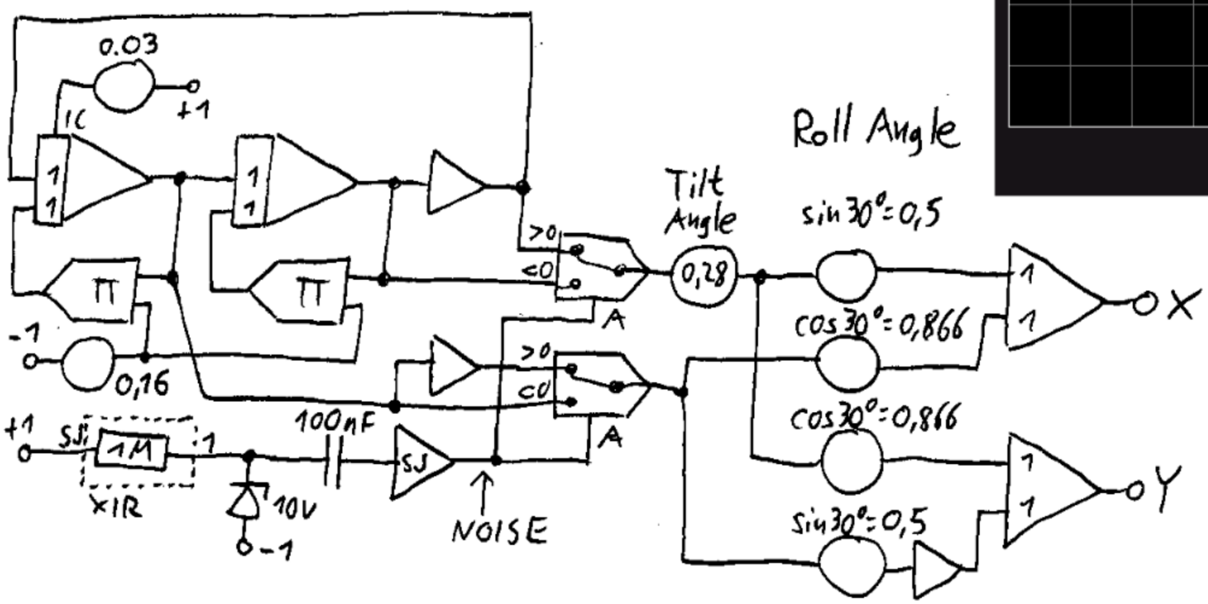
These are two logarithmic spirals, multiplexed by a noise signal.





Let's add a rotation matrix. With the "Tilt Angle" coefficient you can select between edge-on and face-on galaxies:





7.12 Levitating Magnet

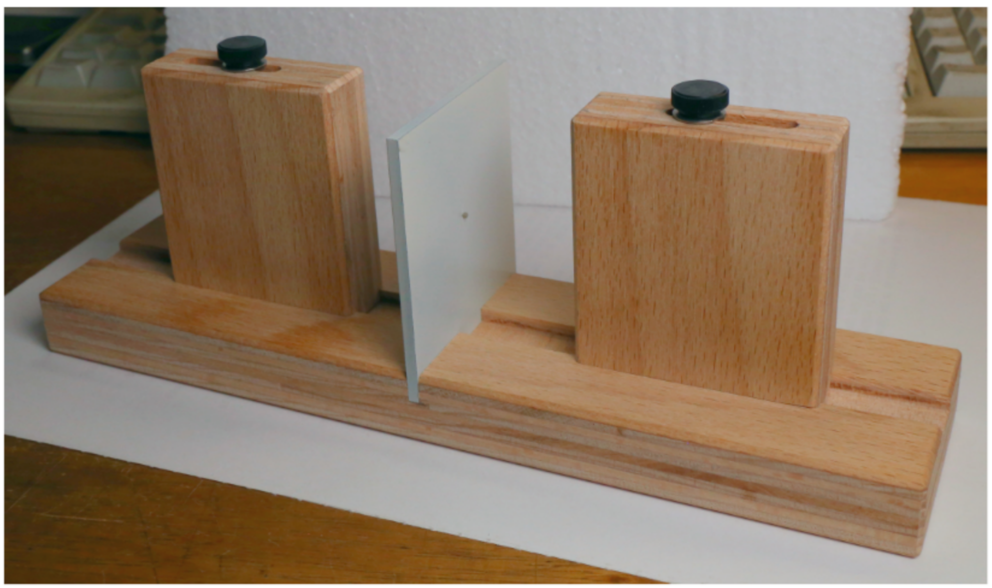
The theoretical background: Earnshaw's theorem states that there can be no arrangement of permanent magnets that allows a small permanent magnet to float stably and freely in space. All conceivable arrangements are unstable. It is only possible if an active control loop is used.

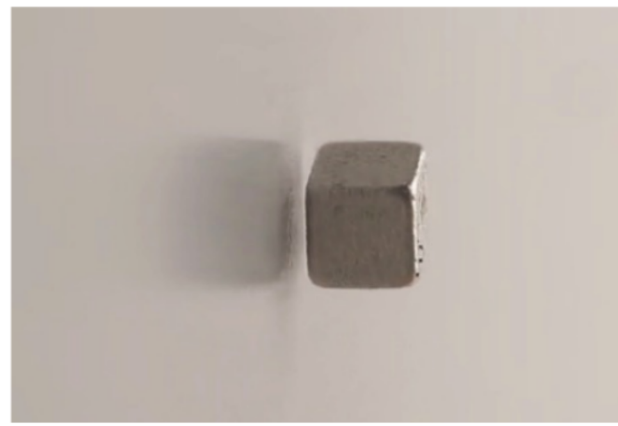
Earnshaw's theorem: https://en.wikipedia.org/wiki/Earnshaw%27s_theorem

This device apparently shows that it's indeed possible to let the magnet float stably in space. There is no active control built in. No current, no electronics, no superconductivity, no low temperature.

This is not my idea. I've only built the device.

The small 1mm^3 cube magnet is barely visible in the center of the white plate:





How does it work?

Two strong permanent magnets are hidden in each of the two wooden blocks. They are cube-shaped with an edge length of 10 mm. The small free-floating magnet has an edge length of 1mm. All three magnets are oriented in such a way that they attract each other. This means that all north poles point to the left and all south poles point to the right. In this arrangement, there is a point exactly in the middle where the attractive forces of the two large magnets on the small magnet compensate each other. This point is stable when the small magnet moves in the plane of the plate. When it is deflected, it is always drawn back to the center of the plate. But this point is unstable when it comes to left/right movement. This means that if the small magnet moves a tiny bit to the left or right (i.e. in the direction of one of the two large magnets), the attractive force of this magnet prevails and the small magnet flies towards this magnet.

Now the plate comes into play. It is made of graphite, which is a diamagnetic material. The small magnet is repelled by the plate, but this force is very small and only acts over a short distance.
The small magnet is not exactly in the middle between the two magnets, but a little to the left of the center. It would therefore fly to the left magnet if this were not prevented by the repulsive force of the plate.
The stable area for the small magnet is very small in the left/right direction, about +-0.2mm. The device only works if the distances are set very precisely. If the small magnet moves too far away from the plate, the attractive force of the right magnet will prevail and it will fly there.

I found the idea in this video (in german): https://www.spektrum.de/.../levitation-bei.../2171757

Does the device violate Earnshaw's theorem? No, because diamagnetic materials aren't allowed in Earnshaw's theorem.

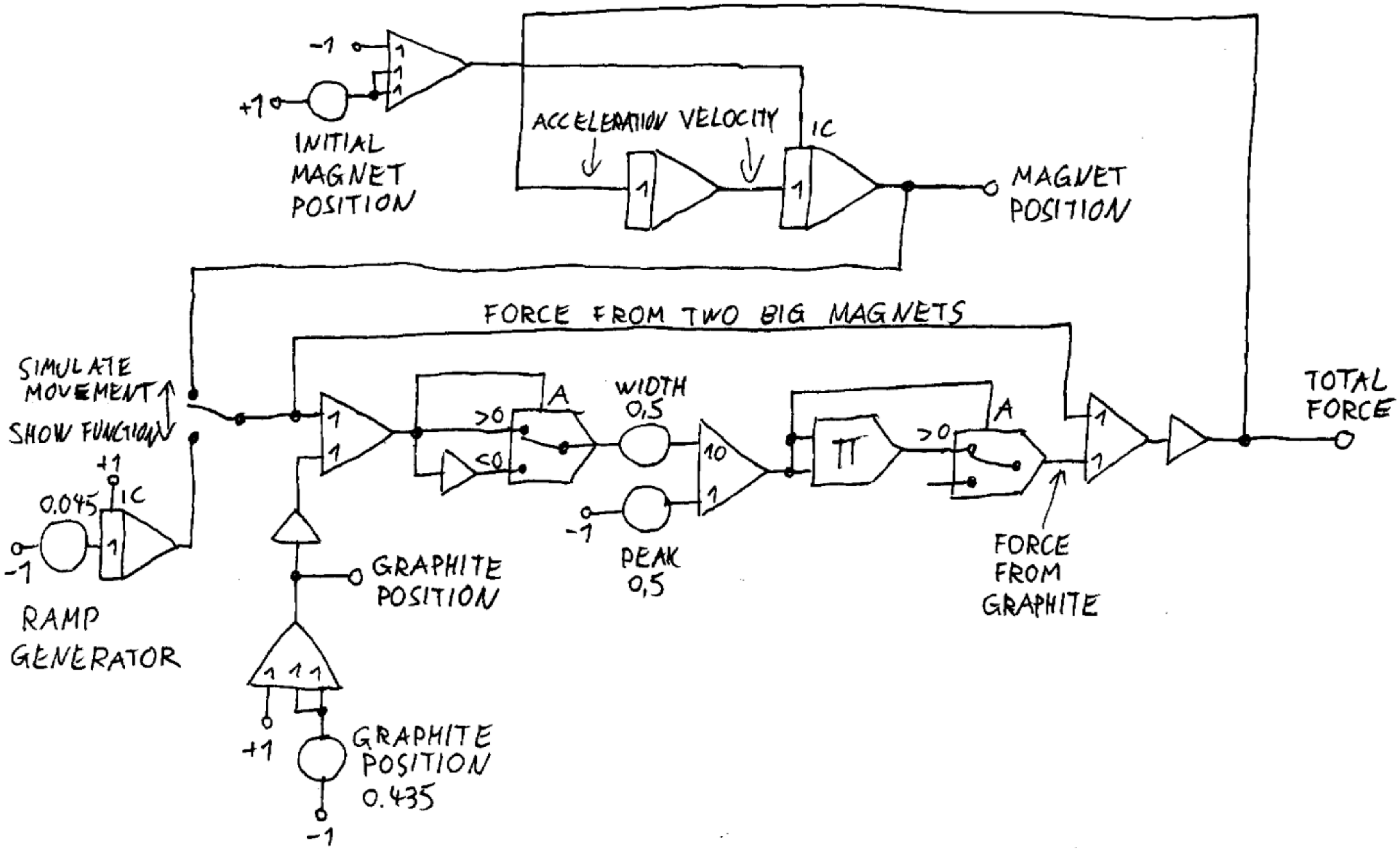
Now let's try to simulate this effect on the analog computer.

The circuit has two integrators for the small magnet's velocity and position.
Position is definded as -1 for the left big magnet and +1 for the right big magnet.
The position is fed into a function which calculates the force which acts on the small magnet.

The small magnet's position can only be stable if this function crosses the zero line with a negative slope.
Without the graphite plate, there is no stable position because the slope is always positive.
A stable position is only possible near the graphite plate, if the magnet is a little bit left of the center, and very close to the plate.

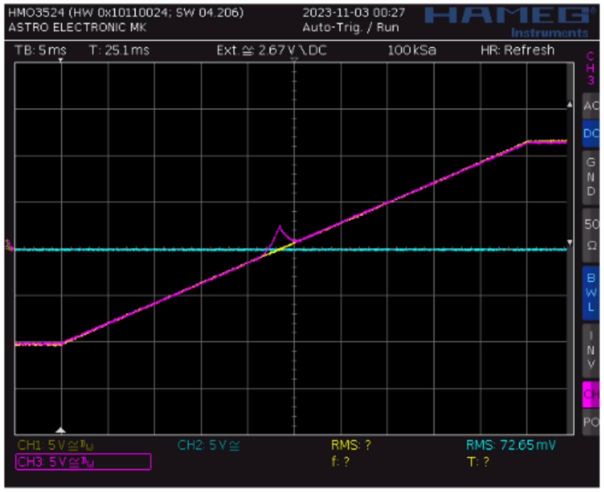
The circuit has a switch where you can select two modes:

- **In upper position the movement of the small magnet is simulated. Use "Magnet Position" as output.**
- **In lower position a ramp generator is connected to the input of the function, so that you can see how the function looks like. Use "Total Force" as output.**

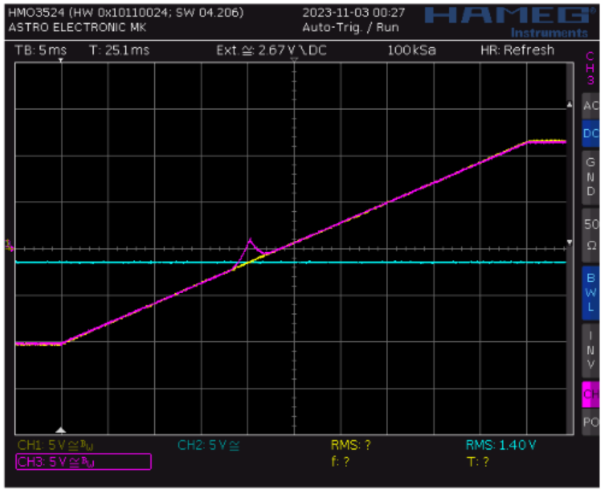

-1
+1
INITIAL MAGNET POSITION
ACCELERATION
VELOCITY
IC
MAGNET POSITION
FORCE FROM TWO BIG MAGNETS
SIMULATE MOVEMENT
SHOW FUNCTION
+1
0,045
IC
-1
RAMP GENERATOR
A
>0
<0
WIDTH 0,5
10
PEAK 0,5
-1
π
A
>0
FORCE FROM GRAPHITE
TOTAL FORCE
GRAPHITE POSITION
+1
GRAPHITE POSITION 0.435
-1

Here are a few examples how the function looks like:

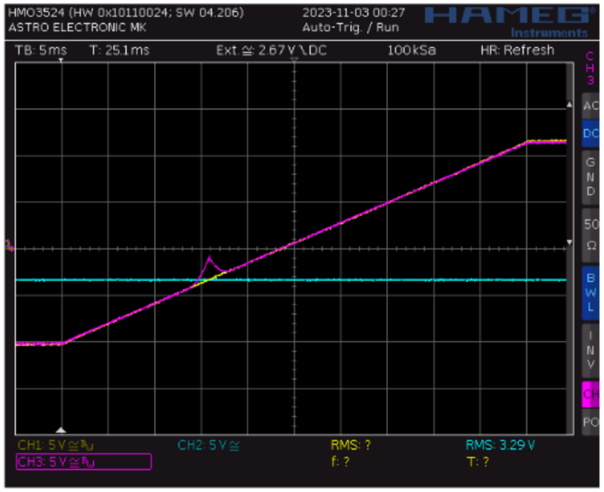
The graphite plate (cyan) is exactly in the center between the two big magnets. The total force (magenta) has a small region with negative slope, but this region doesn't cross the zero line. There is no stable position for the small magnet.



The graphite plate (cyan) is moved a little bit to the left (negative). The total force (magenta) has a small region with negative slope, and this region does cross the zero line. This small region is a stable position for the small magnet.



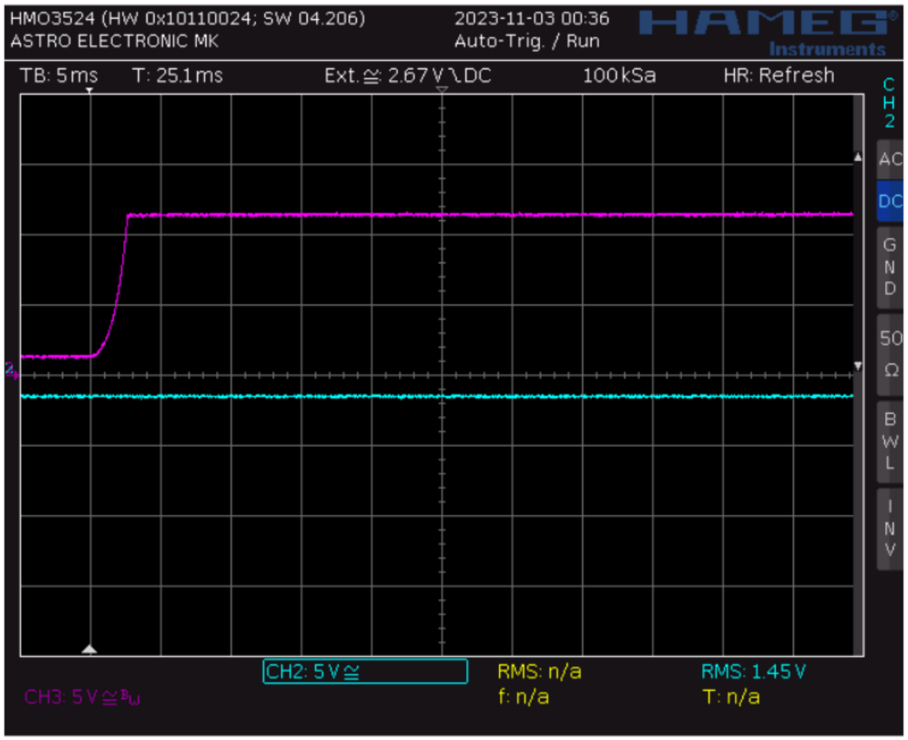
The graphite plate (cyan) is moved even more to the left (negative). The total force (magenta) has a small region with negative slope, but this region doesn't cross the zero line. There is no stable position for the small magnet.



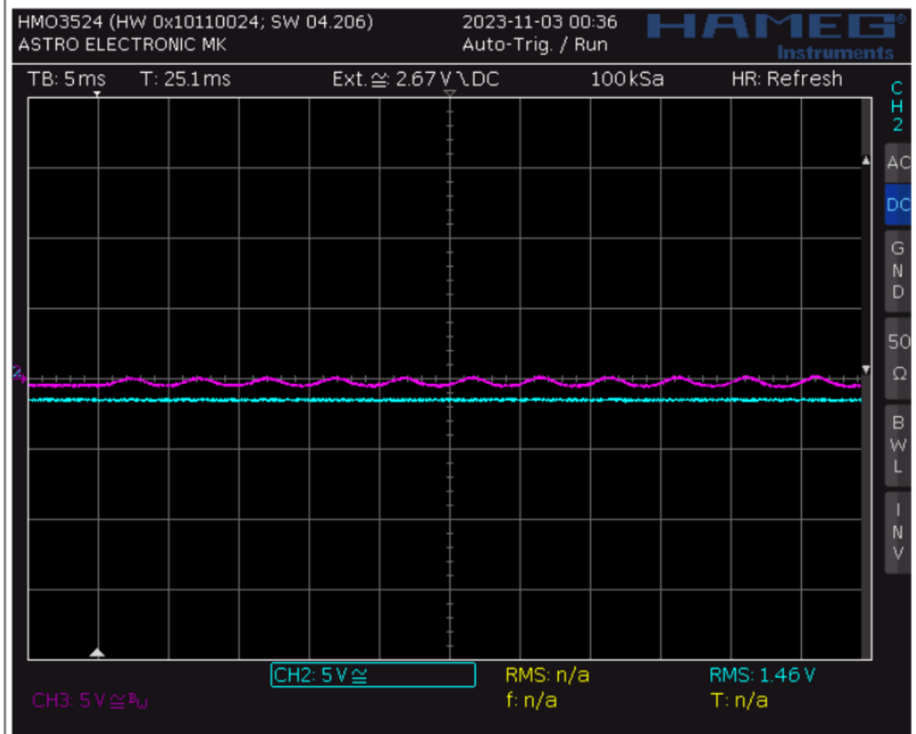
By the way, the circuit is very similar to a tunnel diode oscillator.

Here are two examples for the simulation of the small manget's position over time. In both examples the function (and the graphite plate position) is set as in the middle example on the previous page.

Here the initial position of the small magnet (magenta) is too far away from the graphite plate (cyan), so that the magnet does immediately fly to the right big magnet (positive).



Here the initial position of the small magnet (magenta) is close enough to the graphite plate (cyan), so that the magnet does oscillate around a stable position near the graphite plate.



7.13 Tunnel Diode Oscillator

https://en.wikipedia.org/wiki/Tunnel_diode

AE.1.1			Vergleichsmerkmale von Germanium-Tunneldioden																									
			mechanisch-technologische						elektrische (Kenn- und Grenzwerte bei T_U = 25 °C)																			
			Gehäuse		zus. Kühlung																							
					KS	Kühlblech																						
Typ Kennfarbe Kennzeichen	Her-steller	Bild	Mat	Norm.	KK	Mat	Maße mm cm²	Techno-logie	f_g MHz	f_r MHz	$-r_n$ Ω	$-r_{nstr}$ Ω	r_o Ω	r_{ostr} Ω	l_o nH	c_o *C_G pF	c_{ostr} pF	I_p mA	I_{pstr} mA	U_p mV	I_v mA	U_v mV	$\frac{I_p}{I_v}$	U_{Fp} bei I_p mV	I_{FM} *I_{RM} #i_{FM} mA	P_{tot} mW	T_{jM} *T_{UM} #T_{Umin} °C
TU 4 bl	S	26	K					lg			60	30...100	1,5	<2	1,2	20 *0,5	10...30	1,6	1,3...2	55		250	7		3 *3		*100 #-50
TU 5 ws	S	26	K					lg			90	60...150	2	<3	1,2	10 *0,5	5...20	1,3	0,8...1,6	55		250	7		3 *3		*100 #-50
AE 100	T	69	K					lg			100		1		5	10		1		55	0,15	300				20	
AE 101	T	69	K					lg			100		1		0,5	10		1		55	0,15	300				20	
TU 6 li	S	26	K					lg			130	80...200	3	<6	1,2	5 *0,5	2...10	0,8	0,7...1,2	55		250	7		3 *3		*100 #-50
TU 7 sw	S	26	K					lg			130	80...200	2	<4	1,2	7 *0,5	4...10	1	0,85...1,15	55		250	7		3 *3		*100 #-50
TU 8 li	S	78	K					lg			130	80...200	3	<6	0,75	5 *0,5	2...10	0,8	0,7...1,2	55		250	7		#3		*100 #-50
TU 9 sw	S	78	K					lg			130	80...200	2	<4	0,75	7 *0,5	4...10	1	0,85...1,15	55		250	7		#3		*100 #-50
TU 2 ge	S	26	K					lg			150	<250	1,5	<2	1,2	30 *0,5	<50	1	0,6...1,4	55		250	7		3 *3		*100 #-50
TU 3 rt	S	26	K					lg			150	<250	2	<3	1,2	15 *0,5	<25	0,5	0,25...0,75	55		250	7		3 *3		*100 #-50
AEY 12	SEL	30	M	TO-18				lg	980	200	20		0,4		6	90		15	13,5...16,5	55	1,8	310	8	490	150 *150		75
AEY 11	SEL	30	M	TO-18				lg	1050	310	25		1		6	30		5	4,5...5,5	55	0,7	300	7	480	50 *50		75
JK 30A	SEL	55	K					lg	1100	900	25		0,9		1	30		5	4,5...5,5	60	0,7	320	7	480	50 *50		75
JK 60A	SEL	56	K		=2xJK 30A,Paar																						

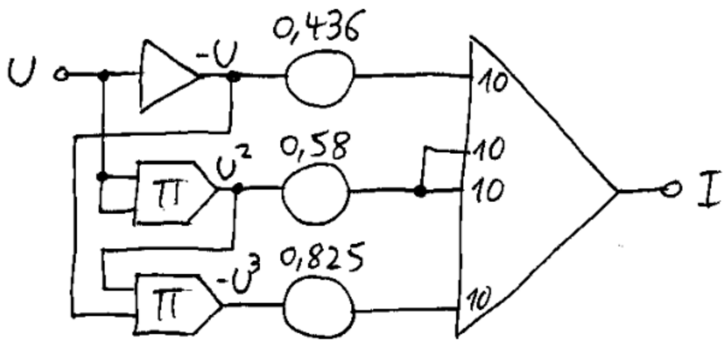
Beispiele von Tunneldioden

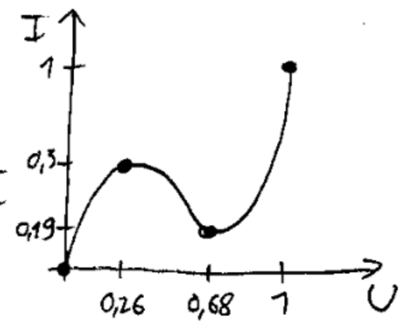
Tabelle II

Typ	Art	Hersteller	Strom und Spannung im Höcker		Tal		Höcker/Tal Strom-verhältn.	Negativer Widerstand	Kapazität	Eigen-induktivität
			mA	mV	mA	mV		Ω	pF	nH
1N2939A	Ge	General Electric	1	60	0,1	350	10	150	4...5	6
1N3150	Ge	General Electric	22	60	0,9	350	8	10	60	6
TU10/1	Ge	Siemens	1	55	0,14	250	7	110	2	0,75
TU14/1	Ge	Siemens	20	55	3	250	7	10	30	1,5
JK19A	Ge	St. El. & C.	1	55	0,16	290	6	110	10	8
JK21A	Ge	St. El. & C.	15	55	2,5	310	6	8	80	8
AE100	Ge	Telefunken	1	55	0,15	300	7	100	10	5
A650	Ga-As	Texas Instruments	10		0,67		15		25	

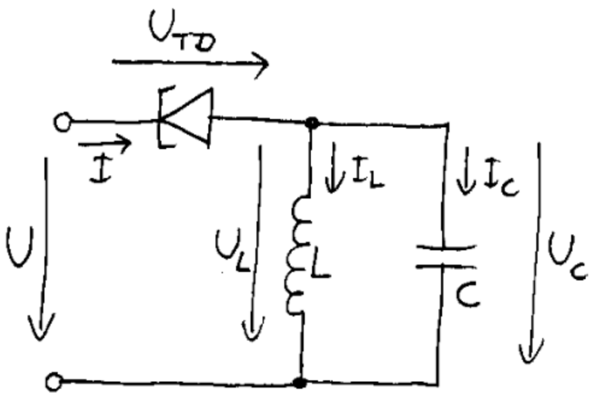
Current over voltage characteristic of tunnel diode (this is a very rough approximation):

$$I = 8{,}25\,U^3 - 11{,}6\,U^2 + 4{,}36\,U$$





Tunnel diode oscillator with equations:



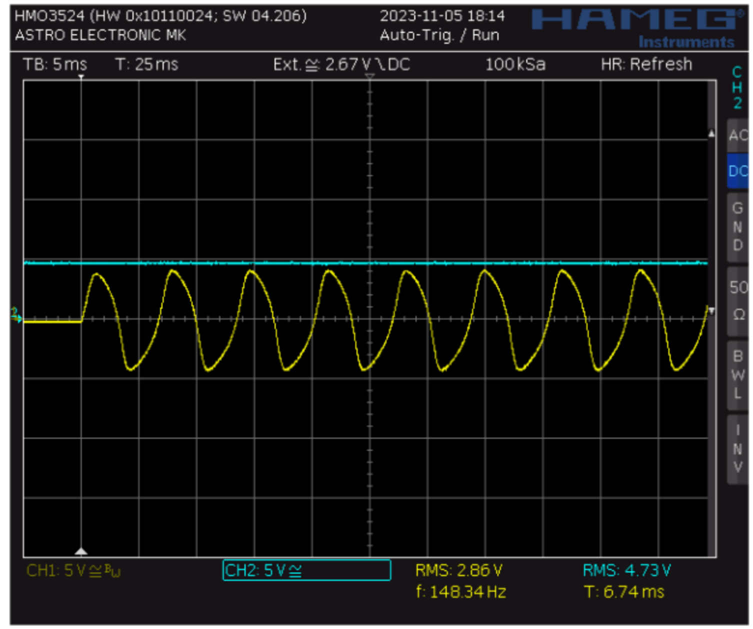
$$\frac{dU_C}{dt} = \frac{I_C}{C}$$

$$\frac{dI_L}{dt} = \frac{U_L}{L}$$

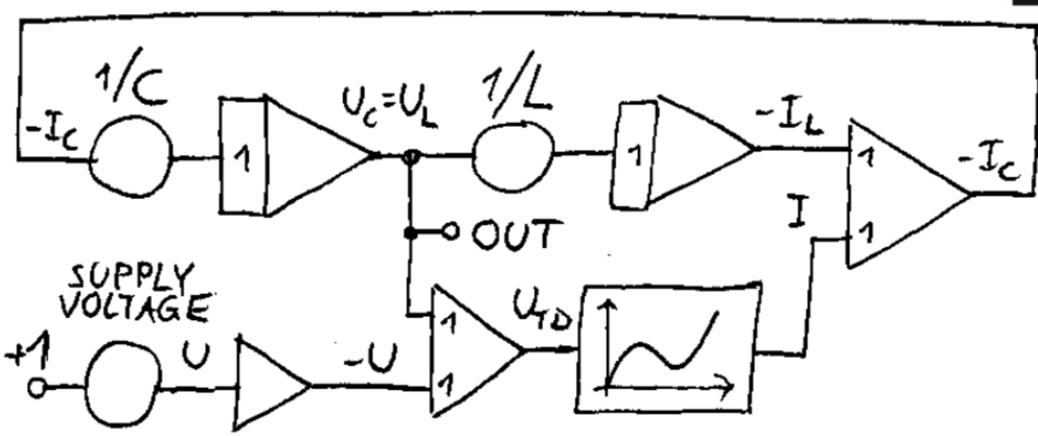
$$I = I_L + I_C$$

$$U = U_{TD} + U_C$$

$$U_C = U_L$$

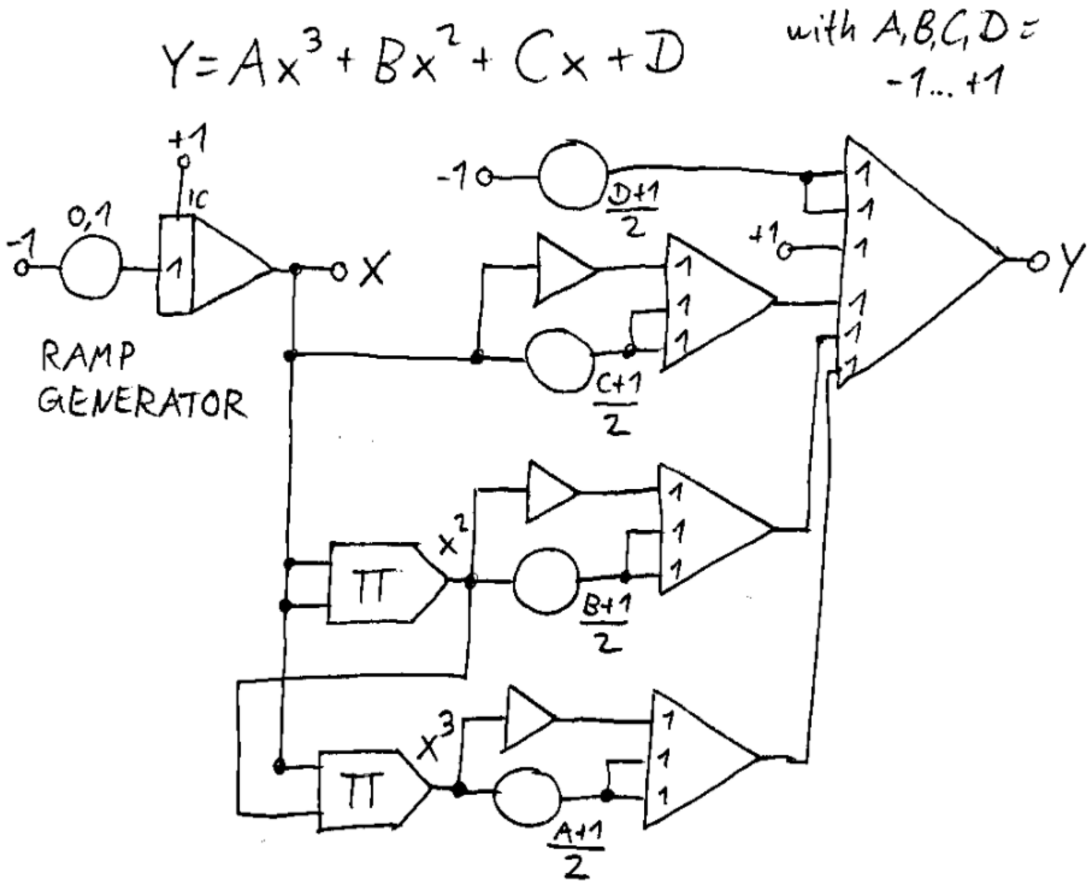


Circuit for tunnel diode oscillator:



7.14 Polynomial Generator y = ax^3 + bx^2 + cx + d

The parameters A, B, C and D are adjustable from -1 to +1.



7.15 Tritone Generator

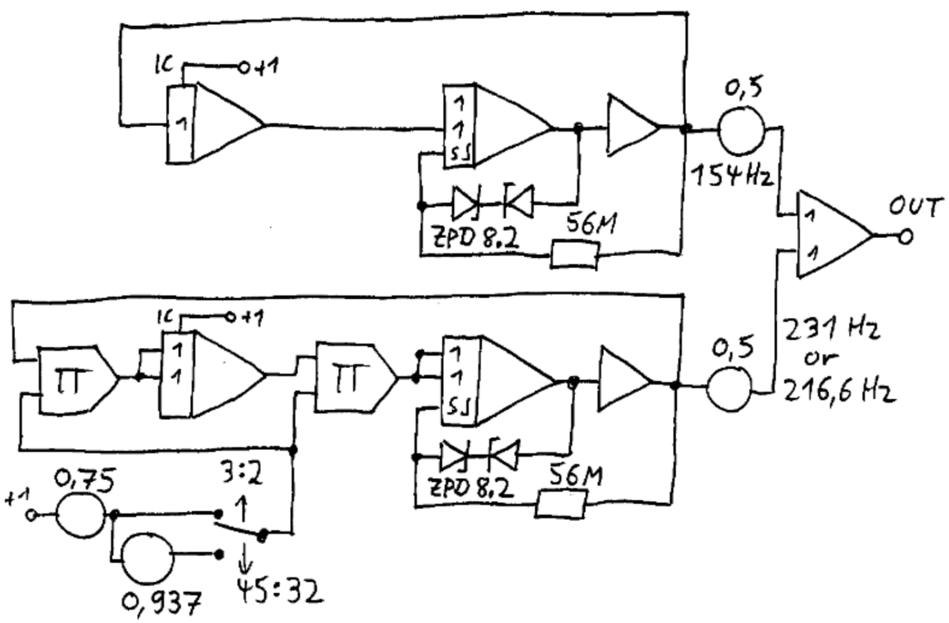
I found it in this book: Michael S. Horn, Melanie West, Cameron Roberts: Introduction to Digital Music with Python Programming, page 62

A tritone in music is defined as a 45:32 interval (or √2:1 interval) which was frequently avoided in composition for its dissonant qualities.

See also: https://en.wikipedia.org/wiki/Tritone

However a 3:2 interval sounds harmonic and is called a "Perfect Fifth". Which means it is the interval between the base note and the fifth note of a diatonic scale on that base note, and it is perfect because it is a 'pure' 3/2 ratio, not the 'impure' $2^{(7/12)}$ = 1.498 interval you find on a piano.

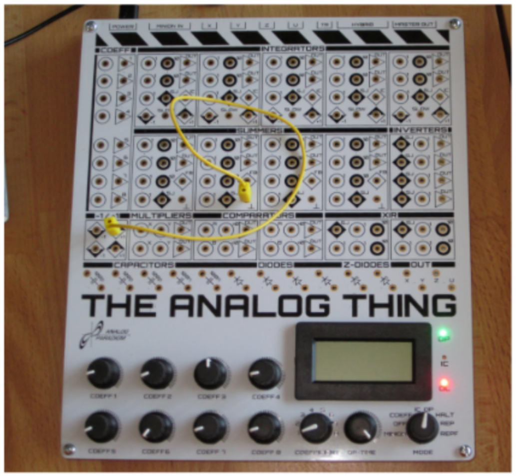
**This circuit simulates both intervals, selectable by a switch. Tune the 0.75 coefficient for exact 3:2 ratio, which can easily be checked with an oscilloscope. The other coefficient is (45/32) / (3/2) = 0.9375. Connect the output to an audio amplifier.
For the exact √2:1 interval set the second coefficient to 0.943.**



8 Problems and Solutions

8.1 Display is off

If you ever have the problem that nothing is shown in the display, you should check if you have accidentally shorted the +1 signal to ground. The same thing may also happen if two or more outputs of summers or integrators are shorted.



8.2 Unexpected Overflows

I had some unexpected overflows when I tested a circuit. I did check with an oscilloscope that all outputs of the computing elememts were in the allowed range. But nevertheless after a few seconds in "OP" mode the overload LED was going on. Finally I figured out that the overload came from an unused integrator. Keep that in mind when you do a calculation in "OP" mode over a longer time. All unused integrators must be disabled by connecting their input to their output.

8.3 Integrators are drifting away too fast

If you ever have the problem that one if the integrators is drifting too fast into overload, it's time to clean the surface of the front panel with alcohol. In my case the overload came within 0.2 seconds. There must have been a fingerprint on the front panel. 200 MΩ * 1nF = 0.2s. After cleaning, it works again as expected.

8.4 THAT doesn't run after power-on

That's a known problem when THAT is in REPF mode during power-on. Switch to REP and back to REPF, then it will run.

9 Unsolved Problems

- Understand how the limiters in figures 5.20 and 5.21 in Bernd's book work.
- Understand problem variables / machine variables / time scaling. Very complicated.

10 Mathematics

10.1 Books about Mathematics

Calculus made easy (written in 1910): http://calculusmadeeasy.org/

Bronstein-Semendjajew: Taschenbuch der Mathematik, 21. Auflage, ISBN 3-87144-492-8

Feldmann, Dietrich: Repetitorium der Ingenieur-Mathematik, ISBN 3-923923-00-7

Feldmann, Dietrich: Repetitorium der Ingenieur-Mathematik, Teil 2, ISBN 3-923923-05-8

Mühlbach, Günter: Repetitorium der Ingenieur-Mathematik, Teil 3, ISBN 3-923923-30-9

10.2 Powers and Roots

$a^x = e^{x \ln(a)}$ $\qquad$ $a^{-x} = 1 / a^x$ $\qquad$ $(1 / a)^{-x} = a^x$ $\qquad$ $a^0 = 1$

$x^{a+b} = x^a * x^b$ $\qquad$ $x^{a-b} = x^a / x^b$ $\qquad$ $x^{ab} = (x^a)^b = (x^b)^a$ $\qquad$ $x^a = (x^{a / b})^b$ $\qquad$ $x^{a/b} = \sqrt[b]{x^a}$ $\qquad$ $x^{a/2} = \sqrt{x^a}$

$\sqrt{x} = x^{1 / 2}$ $\qquad$ $\sqrt[n]{ab} = \sqrt[n]{a} * \sqrt[n]{b}$ $\qquad$ $\sqrt[n]{a/b} = \sqrt[n]{a} / \sqrt[n]{b}$ $\qquad$ $\sqrt[n]{a} = a^{1 / n}$ $\qquad$ $\sqrt[n]{a^m} = (\sqrt[n]{a})^m = a^{m / n}$

$2 - \sqrt{3} = 1 / (2 + \sqrt{3})$

The expression 2^{3^x} (without parantheses) must be interpreted as $2^{(3^x)}$ and not as $(2^3)^x$. It's better to use parantheses.
See also this german Wikipedia page: https://de.wikipedia.org/wiki/Operatorrangfolge

10.3 Logarithms and Exponential Functions

$\log(a * b) = \log(a) + \log(b)$ $\log(a / b) = \log(a) - \log(b)$ $\log(1 / x) = -\log(x)$

$\log(a^b) = b * \log(a)$ $\log(x^2) = 2 * \log(x)$

$\log(1) = 0$ $\ln(x) = \log_e(x)$

$\log_a(x) = \log_b(x) / \log_b(a)$ $\log_a(x) = \ln(x) / \ln(a)$ $\log_e(x) = \ln(x)$ $\log_a(a^x) = x$

$\log_{a^x}(b^x) = \log(b) / \log(a) = \log_a(b)$ $\log(b^x) / \log(a^x) = \log(b) / \log(a) = \log_a(b)$ --> x doesn't care!

$\log_2 0.125 = -3$ $\log_2 0.25 = -2$ $\log_2 0.5 = -1$ $\log_2 1 = 0$ $\log_2 2 = 1$ $\log_2 4 = 2$ $\log_2 8 = 3$ $\log_2 256 = 8$ $\log_2 65536 = 16$

$\log_{10} 0.001 = -3$ $\log_{10} 0.01 = -2$ $\log_{10} 0.1 = -1$ $\log_{10} 1 = 0$ $\log_{10} 10 = 1$ $\log_{10} 100 = 2$ $\log_{10} 1000 = 3$

$\ln(1 / e^3) = -3$ $\ln(1 / e^2) = -2$ $\ln(1 / e) = -1$ $\ln(1) = 0$ $\ln(e) = 1$ $\ln(e^2) = 2$ $\ln(e^3) = 3$

$e^{\ln(0.1)} \approx e^{-2.303} \approx 0.1$ $e^{-1} \approx 0.3679$ $e^0 = 1$ $e^1 = e \approx 2.718$ $e^2 \approx 7.389$ $e^{\ln(10)} \approx e^{2.303} \approx 10$

$\lim_{n \to \infty} (1 + 1/n)^n = e$

10.4 Imaginary Numbers

$i^2 = -1$ $e^{i\pi} = i^2 = -1$ $\ln(i) = i * \pi / 2$ $e^{i\theta} = \cos(\theta) + i \sin(\theta)$

10.5 Trigonometric Functions

$\sin^2(x) + \cos^2(x) = 1$ $\sin(30°) = \cos(60°) = 0.5$

$\sinh(x) = (e^x - e^{-x}) / 2$ $\cosh(x) = (e^x + e^{-x}) / 2$

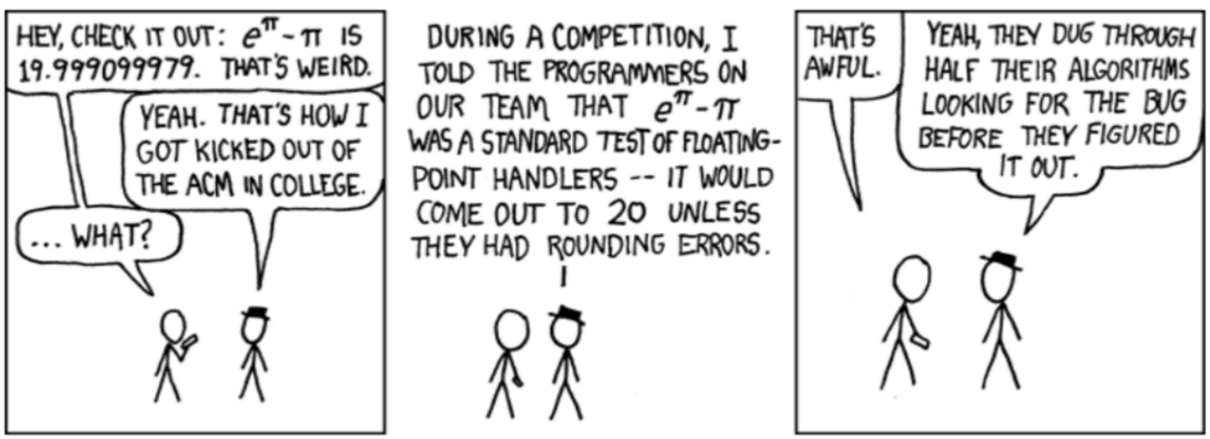
10.6 Approximations

$1 / (1 + a) \approx 1 - a$ For small a

$e \approx 2.718281828...$ $\pi \approx 3.141592654...$

$\sqrt[4]{9^2 + 19^2 / 22} = 3.141592653... \approx \pi$

$e^\pi - \pi = 19.99909998... \approx 20$ https://xkcd.com/217/



10.7 Exact Solutions

10! seconds = 6 weeks

10.8 Geometry

Right triangle: $a^2 + b^2 = c^2$ $h^2 = p * q$

10.9 Divisibility

Divisibility rules: https://en.wikipedia.org/wiki/Divisibility_rule

10.10 CAS (Computer Algebra Systems)

Wolfram Alpha: https://www.wolframalpha.com/

Syntax examples: "series of y=0.006541*2^(10*x) at x=0.421"

GeoGebra: Can be used online: https://www.geogebra.org/cas or as a standalone application: https://www.geogebra.org/download

https://wiki.geogebra.org/en/PointList_Command	**Creates list of points from a list of two-element lists.** **Example: PointList({{1,2},{3,4}})**
https://wiki.geogebra.org/en/FitPoly_Command	**Calculates the regression polynomial of degree *n*.** **Example: FitPoly({(-1, -1), (0, 1), (1, 1), (2, 5)}, 3)**
https://wiki.geogebra.org/en/FitExp_Command	**Calculates the exponential regression curve in the form ae^{bx}.**
https://wiki.geogebra.org/en/FitLog_Command	**Calculates the logarithmic regression curve a + b ln(x).**
https://wiki.geogebra.org/en/FitGrowth_Command	**Calculates a function of the form $a\ b^x$ to the points in the list.**
https://wiki.geogebra.org/en/FitPow_Command	**Calculates the regression curve in the form $a\ x^b$.**
https://wiki.geogebra.org/en/FitSin_Command	**Calculates the regression curve in the form $a + b \sin (c\ x + d)$.**
https://wiki.geogebra.org/en/FitLine_Command	**Calculates the *y* on *x* regression line of the points.**
https://wiki.geogebra.org/en/FitLineX_Command	**Calculates the *x* on *y* regression line of the points.**
https://wiki.geogebra.org/en/FitLogistic_Command	**Calculates the regression curve in the form a / (1 + b e^(-kx)).**
https://wiki.geogebra.org/en/Fit_Command	**Calculates a linear combination of the *functions* that best fit the *points* in the list.**

11 Miscellaneous

11.1 Power Supply

The supply voltage of THAT is 5V, coming from a USB port. The current is about 120mA without any connections, or about 240mA with heavy load at two outputs (the circuit with resolver).

There is a BD2244G-M current limiter at the USB port, which limits the current to 590mA. It doesn't have a protection against wrong polarity. The absolute maximum rating of the input voltage is -0.3V to +7.0V

Data sheet: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/power_switch/bd2244g-m-e.pdf

THAT schematic: https://github.com/anabrid/the-analog-thing/blob/main/ANATHING_OSH/ANATHING-BASE%20-%20Schematic.pdf

11.2 Corristor

https://corristor.com/nonlinear-correlation/

11.3 Hole through Earth

HOLE THROUGH THE EARTH

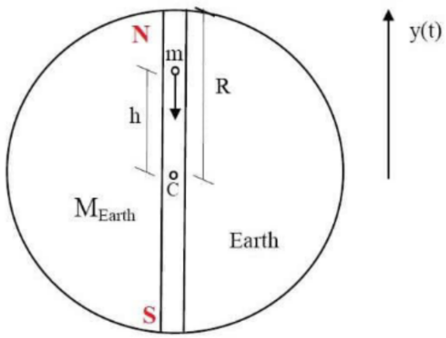


Fig. 1: Mass m falling along a hole drilled through the Earth.

$F = G\frac{mM_h}{h^2} = G\frac{m[\frac{4\pi h^3}{3}\rho]}{h^2} = \frac{4}{3}\pi G m\rho h = -ma$, where: $M_h = \frac{4\pi h^3}{3}\rho$ is the mass of the earth under the falling mass m. So:

$$a = -\frac{4}{3}\pi G\rho h \tag{1.1}$$

Therefore, the falling mass m makes a harmonic motion; in fact, from the (1.1), we have:

$ma = m\frac{d^2y}{dt^2} = -\frac{4}{3}\pi G m\rho y$, from which:

$$m\frac{d^2y}{dt^2} + \frac{4}{3}\pi G m\rho y = 0 \ , \tag{1.2}$$

whose solution is: $y = R\cos\omega t$ (which, by substitution, satisfies the (1.2) indeed, provided that: $\omega = \sqrt{\frac{4\pi G\rho}{3}}$). If the Earth density is $\rho = 5.5\cdot 10^3 kg/m^3$, we have: $T = \frac{2\pi}{\omega} = 5.06\cdot 10^3 s \cong 84\min$ (to and fro); from pole to pole it's 42 min. About the max speed at the centre of the Earth, ($R = 6.3715\cdot 10^6 m$): $V_{Max} = \omega R = 7.9 km/s$.

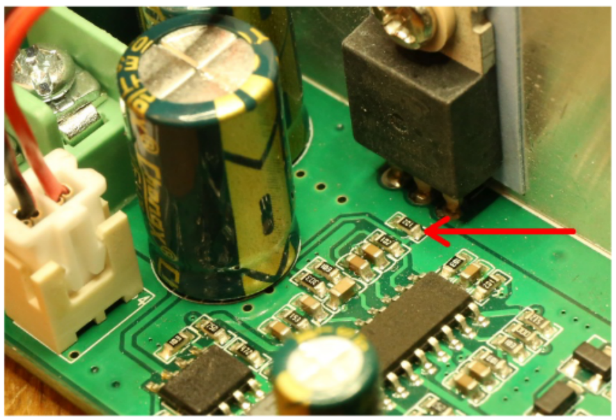
12 Chinese Electronic Modules

SZ-BK0612 400W DC-DC Step-Down Buck Converter

https://de.aliexpress.com/item/1005004398548528.html
60mm x 60mm x 45mm
Input voltage: 10.5 V - 60 V
Output voltage: 0.15 V - 45 V (Maximum output voltage = 0.8 * Input voltage)
Output current: 0.2-15 A (forced cooling is required above 12 A)
Maximum output power: 400 W (with forced cooling)
Continoous power: 280 W (with normal cooling)
Continuous output current: 12A (with normal cooling)
Input undervoltage protection: Yes, if Input < 10.5 V
Input overvoltage protection: No.
Switching frequency: 100 kHz
Efficiency: up to 97%
Short circuit protection: yes, constant current
One end of the voltage and current potentiometers is unused. That are only variable resistors.

RV2 [kΩ]	0	5	10	15	20	25	30	35	40	45	50	55	60	65	65.9
Voltage [V]	0.15	2.39	4.64	6.90	9.15	11.41	13.66	15.90	18.16	20.4	22.6	24.9	27.1	29.4	30.0

RV1 [kΩ]	0	5	10	15	20	25	30	35	40	50	60	70	80	90
Current [A]	0.08	2.58	4.47	5.95	7.15	8.10	8.86	9.50	10.18	11.20	12.05	12.55	13.03	13.43



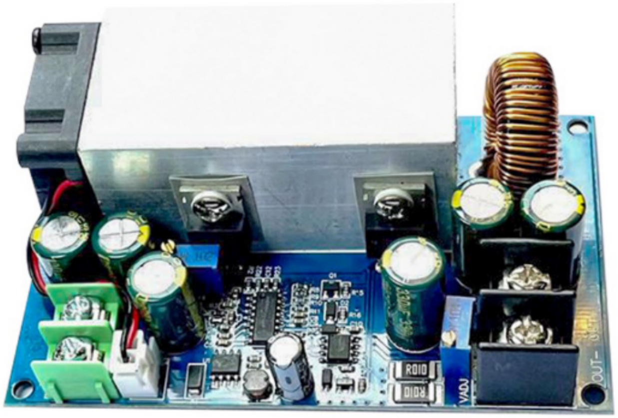
600W DC-DC Step-Down Buck Converter **It has no type number printed on the module.**

https://de.aliexpress.com/item/1005005240989258.html
100mm x 60mm x 35mm
Input voltage: 12 V - 75 V
Output voltage: 2.5 V - 60 V (Maximum output voltage = 0.8 * Input voltage)
Output current: 25 A
Maximum output power: 600 W (with forced cooling)
Continous power: 500 W (with normal cooling)
Input undervoltage protection: No.
Input overvoltage protection: No.
Switching frequency: 80 kHz
Efficiency: 95%
Short circuit protection: yes, constant current, but the current limit is not adjustable.

The type numbers of all three SO chips are ground off.

For an adjustable current limit, replace this 1 kΩ resistor (red arrow) by a 1 kΩ potentiometer. The left side is connected to IN-. The potentiometer value is approximately 37 Ω / A.

There exists also a module with an adjustable current limit, see below.



600W DC-DC Step-Down Buck Converter   Type number SZ-8025CCCCV or GXJA0758-001 ?

https://de.aliexpress.com/item/1005005440291920.html

This is almost the same module as above, but it has an adjustable current limit 1 A - 25 A.
Problem: The 1 kΩ potentiometer for current adjustment is very close to the power transistor.

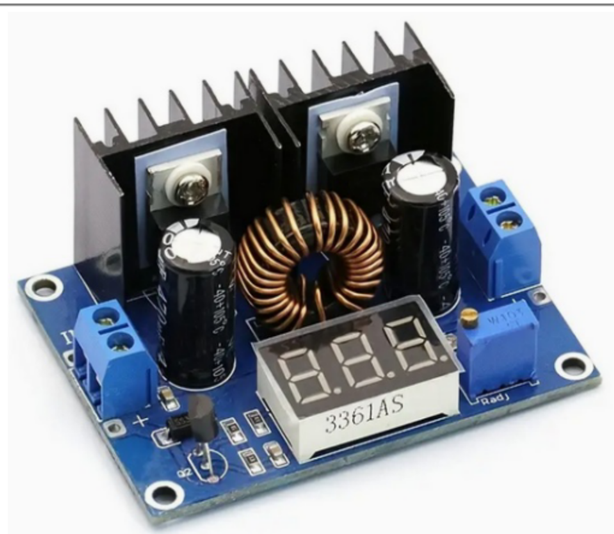
The type numbers of all three SO chips are ground off.

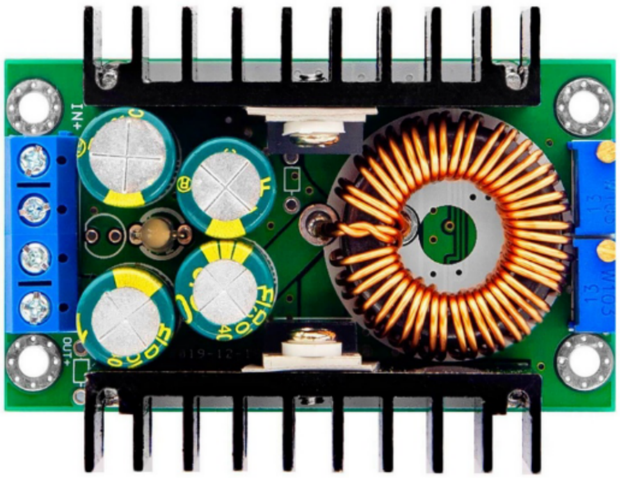
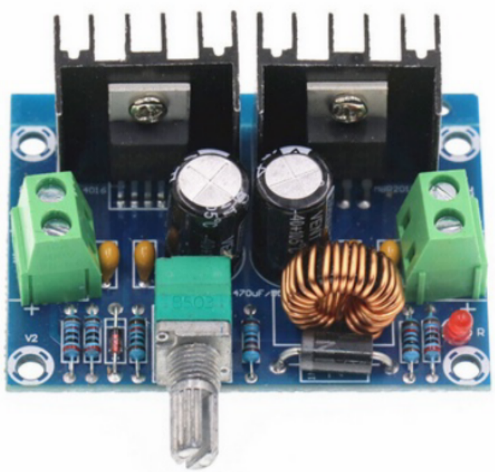
Potentiometer RV2 [Ω]	42.5	77	181	367	557
Current limit [A]	1	2	5	10	15

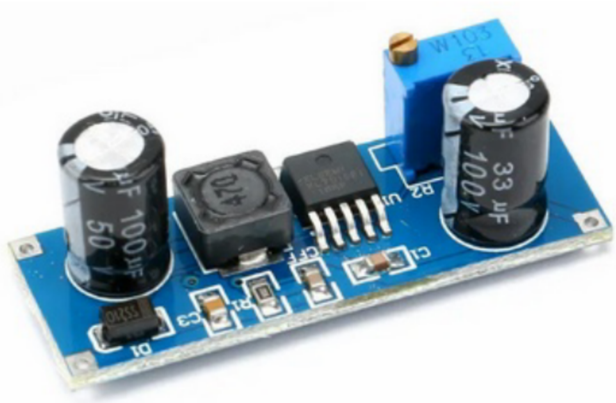
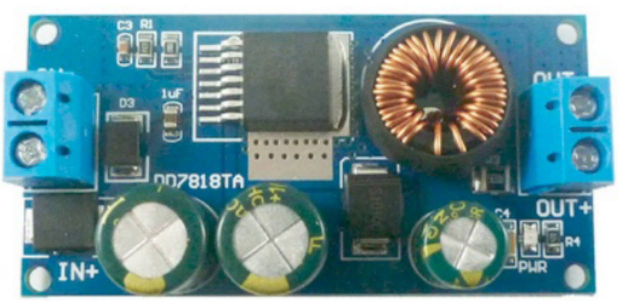
The potentiometer value is approximately 37 Ω / A.

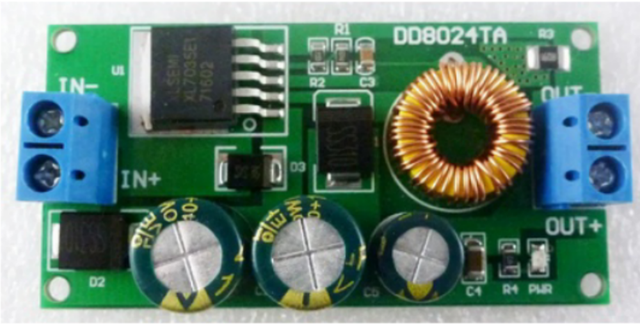
LM2596S DC-DC Step-Down Module

https://de.aliexpress.com/item/1005001621899005.html
46mm x 23mm x 10mm
Input voltage: 7V - 35V
Output voltage: 1.25 V - 30 V
Difference between input and output voltage: min. 1.5V
Output current: maximum 3A
Efficiency: max. 92%
Switching frequency: 150kHz
Left potentiometer is voltage, middle potentiometer is switching point for "charge" LED, right potentiometer is current limit.
Marking on PCB: XW026FR4

	**LM2596 DC-DC Step-Down Module with Voltmeter** https://www.ebay.de/itm/125248503976 **61mm * 34mm * 12mm** **Input voltage: 5V - 40V** **Output voltage: 1.5V - 35 V** **Input current: 4A (Max)** **Output current: 2A (Max)** **Protected against wrong polarity and short circuit** **The voltmeter can measure the input or output voltage, or can be switched off.**
	**XH-M404 DC 4–40 V 8 A Module with Voltmeter** https://www.temu.com/goods.html?_bg_fs=1&goods_id=601099513527641 **Input voltage: 4V - 40V** **Output voltage: 1.25 - 36 V** **Maximum current: 8A (5A continuous)** **Maxumun power: 200W** **Efficiency: 94%** **Short circuit protection, input polarity protection, overtemperature protection.**

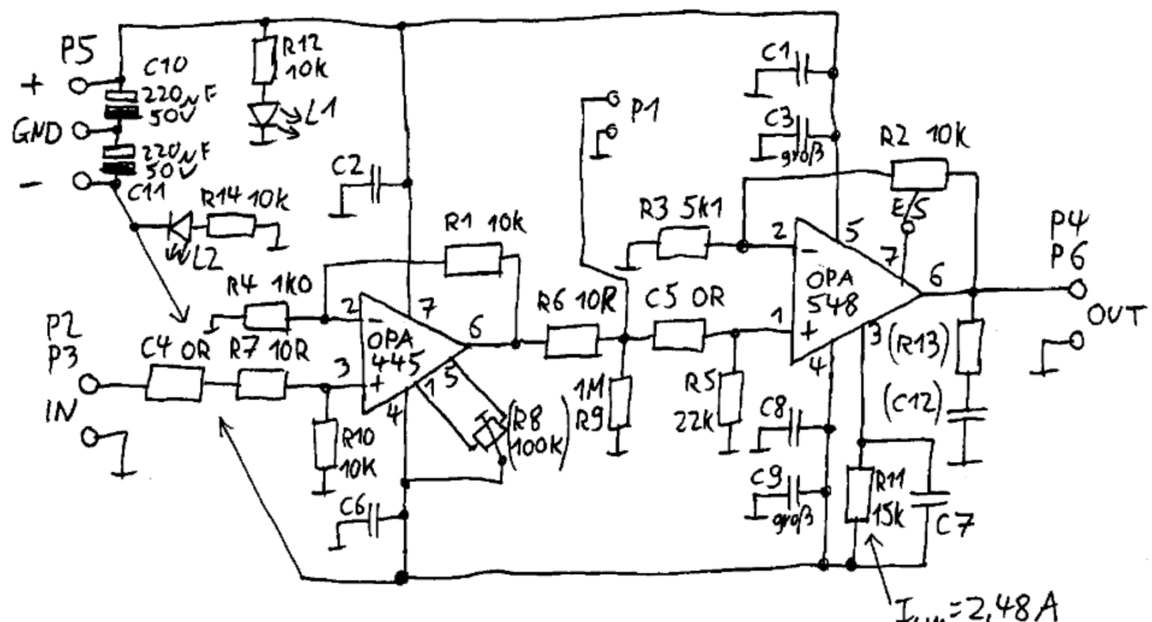
	XL4016 Step Down Buck Converter https://www.ebay.de/itm/144127302529 65mm x 48mm x 24mm Input voltage: 5 - 40V Output voltage: 1.2 - 35V Output current: 4A (forced cooling is required above 65°C) Adjustable current limit: 0.2A - 4.0A Output power: up to 140W
	8A DC-DC Step-Down Module XL4016 https://www.ebay.de/itm/175521917179 61mm * 41mm * 27mm Input voltage: 4V - 40V Output voltage: 1.25V - 36V Model number: XH-M401 Maximum output current: 8A (5A continuous, forced cooling above 5A) Maximum power: 200W Efficiency: max. 94% Switching frequency: 180 kHz Problem: The voltage potentiometer might be negative logarithmic? It's difficult to set a small output voltage.

	**XL7015 DC-DC Converter Step-down Module** https://de.aliexpress.com/item/1005004584302304.html Input voltage: 5 V - 80 V Output voltage: 5V - 20V Maximum Output current: 0.8 A Maximum output power: 7 W 44 mm x 16 mm XL7015E1 datasheet: https://pdf1.alldatasheetde.com/datasheet-pdf/view/1134399/XLSEMI/XL7015E1.html
	**DC-DC Converter Module DD7818TA** https://de.aliexpress.com/item/1005003141428190.html Input voltage: 8 V - 80 V Output voltage: Different modules are available from 3.3 V to 15 V Output current: 2.1 A 60 mm x 30 mm x 16 mm

	**DC-DC Converter Module DD8024TA** https://de.aliexpress.com/item/1005005878674288.html **Input voltage: up to 80 V** **Output voltage: Different modules are available from 3.3 V to 15 V** **Output current: 1 A** **59 mm x 30 mm x 16 mm**

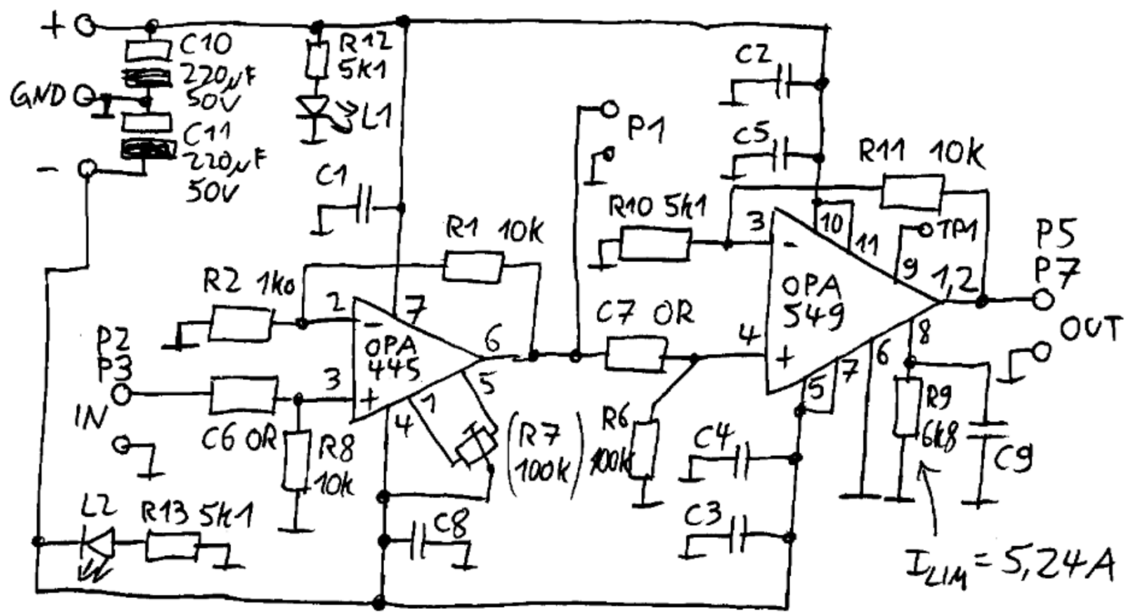
OPA548 Power-OP-Amp Module

https://de.aliexpress.com/item/32959444043.html
https://de.aliexpress.com/item/1005005046010895.htm
96mm x 64mm x 42mm
Supply voltage: +-10V ... +-30V (OPA548 works with +-4V, but OPA445 needs +-10V, replace by OPA551 for +-4V supply)
Continuous output current: 3A
Peak output current: 5A
Gain-bandwidth product: 1MHz
Has a OPA445 chip as pre-amplifier.
First stage gain: R1 / R4 + 1 = 11 non-inverting
Second stage gain: R2 / R3 + 1 = 2.96 non-inverting
Total gain: 32.57 non-inverting

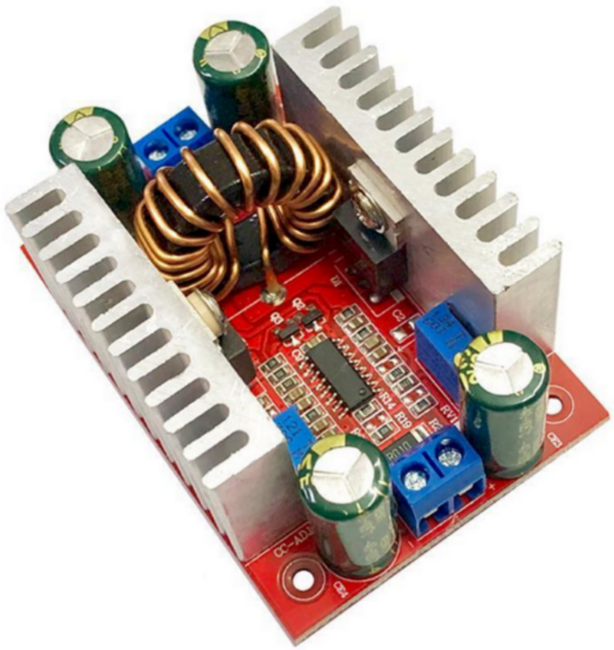
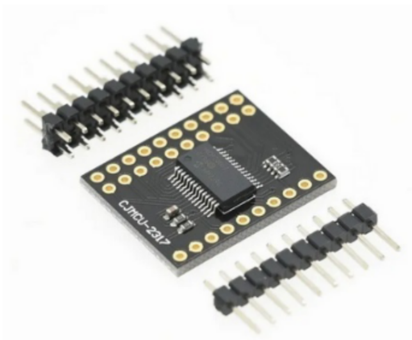


OPA549 Power-OP-Amp Module

https://de.aliexpress.com/item/1005001876749797.html
https://de.aliexpress.com/item/1005005046010895.html
86mm x 81mm x 52mm
Supply voltage: +-10V ... +-30V
It would work from +-4V to +-30V, if the preamplifier OPA445 is replaced by OPA551
Continuous output current: 8A Peak output current: 10A
Gain-bandwidth product: 900kHz Input impedance: 10 kOhm Input voltage: 2Vpp (max)
Output voltate: 54 Vpp (max) with +-30 V supply Current limit is set to 5.24 A by default
Output power: 100 W (max)
Gain: first stage 11, second stage 3, total 33, g1 = r1/r2 + 1, g2 = r11/r10 + 1
Input offset voltage: 1mv (max) Input offset drift: 2 uV/℃ (max)
Problem: There is an air gap between the OPA549 and the heatsink, and when you correct this by adjusting the heatsink position, then two of the mounting nuts are very close to the heatsink.



	**OPA549 Power-OP-Amp Module with large Heatsink (there may be two different version available, constant voltage and constant current, I'm not sure)** https://de.aliexpress.com/item/1005005957798467.html https://de.aliexpress.com/item/1005005790167035.html https://de.aliexpress.com/item/1005004654749888.html https://de.aliexpress.com/item/1005005957420987.html https://de.aliexpress.com/item/1005003740927088.html

	**400W Step-Up Module** **Input voltage: 8.5 V to 50 V, no protection for wrong input polarity** **Input current: max. 15 A (has a fuse on the board)** **Output voltage: 10 V to 60 V** **Output current limiter: 0.2 A to 12 A (forced cooling required above 7A)** **Switching frequency: 150 kHz** **Dimensions: 67 x 48 x 28 mm** **Left potentiometer in picture is current limit, right potentiometer is voltage** **It seems the current limit can't be set with shorted output.** **The chip is TL494C:** https://www.ti.com/lit/ds/symlink/tl494.pdf https://www.ebay.de/itm/386264182944 https://de.aliexpress.com/item/1005003041306288.html
	**Bidirectional 16-bit serial / parallel I2C SPI interface MCP23017** https://de.aliexpress.com/item/1005006386066328.html **MCP23017 has I2C interface, also available in DIP28 package** **MCP23S17 has SPI interface, also available in DIP28 package (but quite expensive)** **Data sheet:** https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP23017-Data-Sheet-DS20001952.pdf

Index