Electrical Engineering - Programmable Electronics

Jean-Louis Noullet - ENAC - 2024

1. Introduction

1.1 Purpose of this document

- Context : This is a support document for a 30-hour practical training in electronics for beginners, focused on low-power, low-voltage, low-frequency experiments, involving:
    - simulation of circuits
    - wiring of discrete components
    - measurement of signals with an oscilloscope
    - development of embedded software on a microcontroller.
- Audience : The reader is not a specialist of electrical engineering or electronics, but is supposed to have a basic "academic" knowledge of:
    - the laws of physics in general, electricity in particular
    - associated mathematics like differential equations
    - programming, in particular the C language
- Contents : The document does not describe the experiments to be done (this is provided separately), does not repeat "well known theories" (which can be better found on Wikipedia), but rather gives clues on "putting things together".

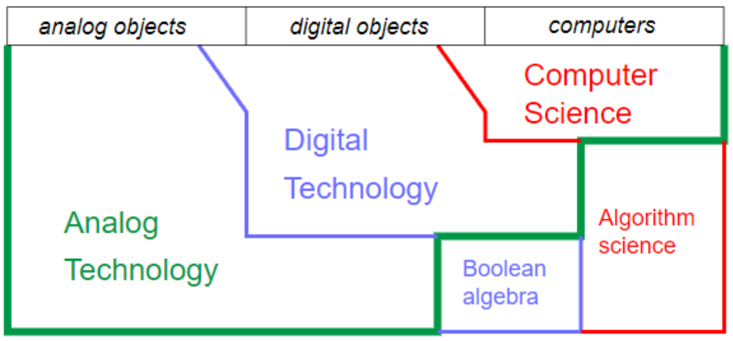
1.2 Analog vs Digital

Consumers easily identify analog objects, digital objects and computers, without caring much about the underlying technologies

A superficial vision associates analog objects with old-style, respected vintage or outdated technologies.

The clocks example leads to the observation that analog deals with continuous variables (like the angular position of the clock hands) while digital manipulates discrete data like numbers and text. That is a deeper vision.

In engineering, the analog technology deals with continuous quantities like voltages and currents and with electronic components like transistors and capacitors, understood with the help of physics theory.



The digital technology deals with logic gates and sequential blocks where the digital binary information is carried by voltages and currents, with the help of transistors and other analog devices.

So digital technology is not built beside the analog technology but rather on top of it, with the help of additional theoretical constructs.

Likewise, computer science is built on top of digital and analog technologies.

The top surface of this diagram is what people perceive. The angled lines express the fact that many "analog objects" contain digital components (the quartz clock is an example), and many "digital objects" actually contain an embedded computer (or more).

What is the origin of the word "analog" in this context ?

Before the advent of digital technology, people were doing electronics without naming it "analog".
In order to solve engineering problems which were too hard to solve manually, engineers invented analog computers which are based on mathematical analogies between the problem to solve and the physics of the computing device.
First analog computers were mechanical or electromechanical, and from 1950 electronic analog computers came in use, and were superseded by digital computers only after 1980.

In analog computers, problem parameters and variables were represented by continuous physical quantities, not numbers, this is still understood as the formal difference between analog and digital systems.

2. The Analogies

The most complete analogies between electricity and other domains of physics are those which extend to systems represented by **differential equations of the second degree**. But some equations limited to the first degree are useful too.

To establish an analogy, one has to select **two fundamental quantities similar to voltage and current**. It his helpful but not mandatory to select one which represent a potential (some capability of action depending on the position in space, which exists event while the system is idle), and the other which represents a flow, something in motion

In electricity, the quantity which is commonly named voltage is more precisely the "electric potential difference" and in some cases "electromotive force".
It is worth remarking that the french word for voltage is "tension", which means "pull" or "traction", while the chinese word for voltage is 电压 which means "electric pressure".

The scientists who created this vocabulary for electricity were surely inspired by the hydraulic analogy, this is the one to begin with.

2.1 The Hydraulic Analogy

When a mass is lifted to a higher altitude, its gravitational potential is increased, at the cost of some energy.

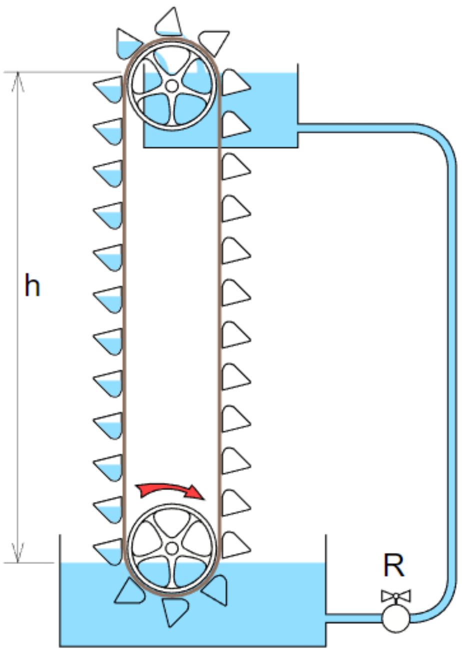
An altitude difference in a fluid like water creates a pressure, which in turn creates a force when applied to a given solid area.

So the analogy between voltage and altitude or pressure, seems very consistent with the vocabulary, coming together with the analogy between electric current and fluid flow

To make the analogy work, we need first to state that the fluid density and the gravitational acceleration are fixed.
Then we may represent the potential as a pressure or as a height, and the flow as a volume flow or a mass flow, these choices are arbitrary.

2.1.1 Legacy analogy

The first table "legacy" is established with height and mass flow as fundamental quantities, with kgf as force unit (apologies to Mr Newton) in order to keep the equations simpler



Hydraulic domain (legacy units)	*Electrical domain*
Pressure (m) h	Voltage (V) V
Flow (kg/s) f	Current (A) I
Quantity of fluid (kg) M = f*t	Electric Charge (C) Q = I*t
Energy (kgf.m) W = M*h = h*f*t	Energy (J) W = V*Q = V*I*t
Power (kgf.m/s) P = h*f = W/t	Power (W) P = V*I = W/t
Restriction R = h/f	Resistor(Ω) R = V/I
Tank "area" (kgf/m) A = M/h	Capacitance (F) C = Q/V = I/(dV/dT)
Inertial pipe j = h/(df/dt)	Inductance (H) L = V/(dI/dt)

The bucket elevator was chosen in the example picture because it demonstrates clearly the transformation of the mecanical energy applied to the lower pulley into potential energy recived by the water.

When the valve R is closed, the relative pressure that it blocks is proportional to the height h, regardless of its own altitude.

If the valve is slightly open, it creates a restriction which acts like a resistor, determining the h/f ratio and converting the h*f power into heat.
This is a little harder to figure out, since the heat is immediateley carried away by the fluid and we do not observe any local temperature increase at the valve itself.

2.1.2 SI analogy

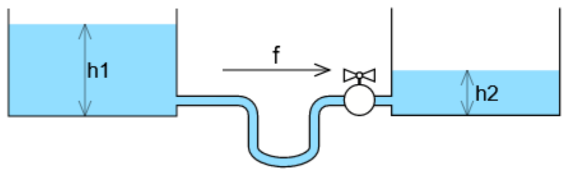
This second table example "SI units" is established with pressure and volume flow as fundamental quantities :

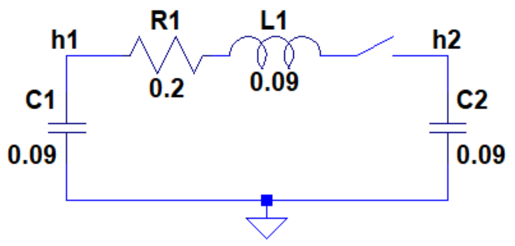
Hydraulic domain (SI units)	*Electrical domain*
Pressure (P) ρ	Voltage (V) V
Flow (m^3/s) f	Current (A) I
Quantity of fluid (m^3) V = f*t	Electric Charge (C) Q = I*t
Energy (J) W = ρ*V = ρ*f*t	Energy (J) W = V*Q = V*I*t
Power (W) P = f/d = W/t	Power (W) P = V*I = W/t
Restriction (Ps/m^3) R = ρ/f	Resistor(Ω) R = V/I
Tank "area" (m^3/P) A = V/ρ	Capacitance (F) C = Q/V = I/(dV/dT)
Inertial pipe (Ps^2/m^3) j = ρ/(df/dt)	Inductance (H) L = V/(dI/dt)

This version of the analogy removes the explicit use of gravity, so it is usable for systems using pumps other than elevators, and capacitive storages using some elastic force instead of gravity.

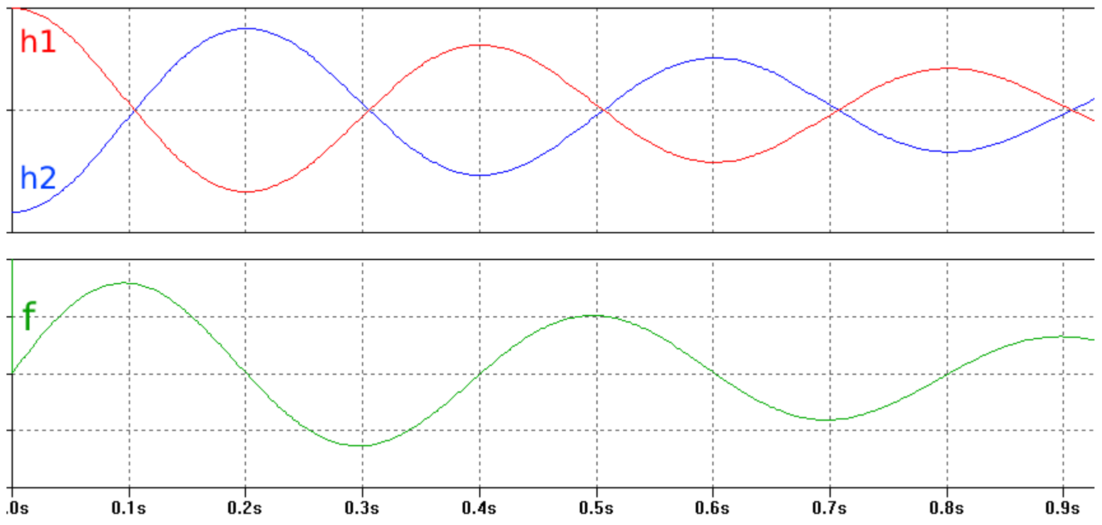
2.1.3 The hydraulic resonator

A resonator or "2nd order system" is made by the combination of two devices that exchange energy, one storing energy by integration of the flow ("static storage"), the other one by integration of the potential ("inertial storage").





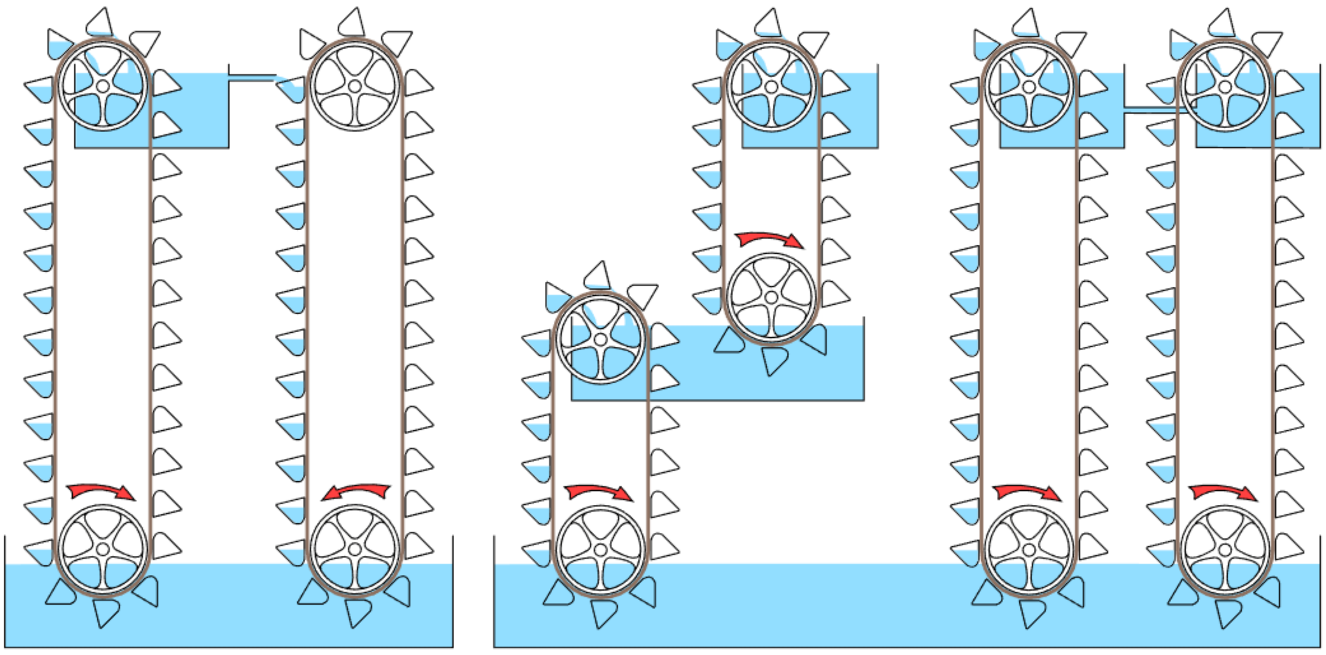
A pair of communicating vessels are the hydraulic analogue of the RLC circuit, where two tanks store energy by gravity ("static storage"), while the accelerated fluid in the pipe stores kinetic energy. The friction in the pipe constitutes the dissipative element which damps the oscillation, like R



With the valve closed, the tank are filled with h1 > h2. Then at time 0 the valve is fully opened. The fluid flows to the right, until h2 = h1. At that point the potential difference is zero, but all the energy is now stored as kinetic energy in the pipe, allowing h2 to become higher than h1, causing a deceleration of the flow, and so on.

Note: a simple U-shaped tube would also present a similar oscillation, in this case there are no separated components for gravitational energy and kinetic energy, the system is said "distributed". This happens as well in electronics, in high frequency systems (like more than 500 Mhz), resonators can have L, R and C merged in a single device.

2.1.4 Other basic hydraulic circuits



From the left to the right:

- a closed circuit: the generator-motor combination :
  - the first bucket elevator behaves like an electric generator: it generates hydraulic power from mechanical power
  - the second one behaves like a motor, it converts hydraulic power back to mechanical power, available at the shaft of its lower pulley (a variant of the well-known water wheel)
- the series combination: the height (or pressure) add, while the flow is common, like in series electric circuits where the voltages add while the current is common
- the parallel combination: the flows add, while the height is common, like in parallel electric circuits where the currents add while the voltage is common

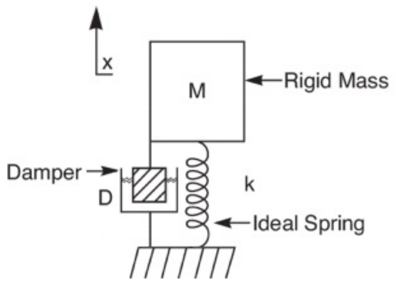
2.2 The Mechanical Analogies

2.2.1 The Mechanical analogy of the first kind

This analogy is close to the hydraulic analogy previously presented, with force replacing pressure and velocity replacing fluid flow.
It is well known from the analogy between the spring-mass resonator and the RLC circuit.

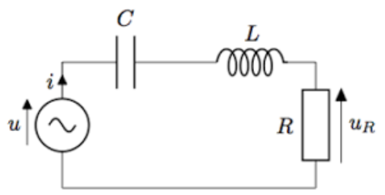
Mechanical domain (first variant)	*Electrical domain*
Force (N) F	Voltage (V) V
Velocity (m/s) v	Current (A) I
Dispacement (m) x = v*t	Electric Charge (C) Q = I*t
Work (J) W = F*x = F*v*t	Energy (J) W = V*Q = V*I*t
Power (W) P = F*v = W/t	Power (W) P = V*I = W/t
Viscous friction (Ns/m) D = F/v	Resistor(Ω) R = V/I
Spring compliance (m/N) 1/k = x/F (k is the stiffness)	Capacitance (F) C = Q/V = I/(dV/dT)
Mass (kg) M = F/(dv/dt) = F/γ (γ is the acceleration)	Inductance (H) L = V/(dI/dt)
Elastic energy (J) W = ½*(1/k)*F^2 = ½*k*x^2	Energy in capacitor (J) W = ½*C*V^2
Kinetic energy (J) W = ½*M*v^2	Energy in inductor (J) W = ½*L*I^2

2.2.2 The mass-spring resonator



Everytime something is vibrating, the spring-mass system is involved. Since every physical system loses energy by some way, we should consider a mass-spring-damper system, the analogue of the RLC circuit

The theoretical diagrams use to represent the damper as piston in a cylinder with some leakage. This comes from the dampers used to shorten unwanted resonance of many mechanical systems, like the suspension of a car.



This damping is called "viscous friction" because it involves a fluid (liquid or gas), providing a linear (proportional) action like its analogue, the resistor. (Solid friction is more difficult to model, being strongly non-linear).
Friction with the air contributes to the damping of every vibration.

When it is not too much damped, the system has a resonating frequency
ω = sqrt(k/M) = sqrt(1/L*C) in rad/s
(divide by 2Π for converting to Hertz)

Note: on the mechanical drawing, the damper and the spring look like they are in parallel. In fact the three elements share the same displacement x (analogue of current) while the forces (analogue of voltage) are added, so this is equivalent to electric elements in series.

2.2.3 The rotational variant

A similar mechanical analogy can be defined for rotating mechanisms :

- force is replaced by torque
- velocity is replaced by rotational speed (angular velocity)
- displacement is replaced by angle
- and so on

A familiar example of a rotational resonator is the balance wheel used in mechanical watches and clocks.
With minimal damping, the frequency of its oscillation is precise enough to be the reference for time measurement. The mechanism called escapement provides just enough energy to compensate the damping caused by friction.
Similarly, LC oscillators are composed of a RLC resonator associated with a feedback circuitry which provides energy to compensate the losses.

2.2.4 The Mechanical analogy of the second kind

The equations of electricity (viewed as a systems described by differential equations of the second degree) have a remarkable symmetry.
The two fundamental quantities V and I have symmetrical roles, like L and C do. This symmetry is sometimes called "duality".

This is formally applicable to the mechanical domain as well, but swapping force and velocity has no pratical interest.
However, describing a second kind of mechanical-electrical analogy where V and I are swapped may make sense.

Mechanical domain (2nd variant)	*Electrical domain*
Velocity (m/s) v	Voltage (V) V
Force (N) F	Current (A) I
Impulse (Ns) I = F*t	Electric Charge (C) Q = I*t
Dispacement (m) x = v*t	Magnetic flux (Wb) Φ such as V = dΦ/dt
Work (J) W = F*v*t	Energy (J) W = V*Q = V*I*t
Power (W) P = F*v = W/t	Power (W) P = V*I = W/t
Viscous friction (m/Ns) d = v/F	Resistor(Ω) R = V/I
Mass (kg) M = F/(dv/dt) = F/γ (γ is the acceleration)	Capacitance (F) C = Q/V = I/(dV/dT)
Spring compliance (m/N) 1/k = x/F (k is the stiffness)	Inductance (H) L = V/(dI/dt)

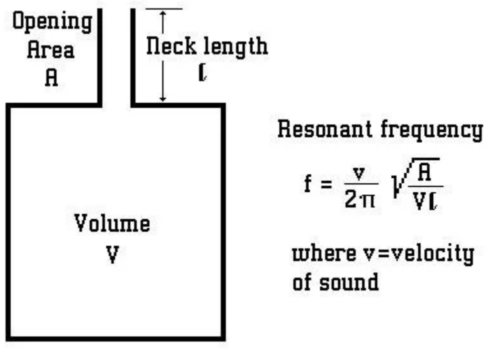
Position or displacement is no longer the analogue of electric charge, but it is the analogue of the integral of voltage over time, which is the magnetic flux, so a line was added for it. (It was omitted in previous tables because it is noticeable only in the inductor.)

This analogy is less intuitive than the first variant, except in the case of systems containing electric motors or other electromagnetic actuators.
An ideal electric motor or generator has a speed proportional to the voltage and a torque proportional to the current, which fits well with this analogy. Notably, an idling electric motor viewed from the electric domain behaves like a capacitor, storing kinetic energy in its rotor

When this analogy was first presented in 1932, a controversy arose because some authors claimed that this new analogy was "more complete" and that the older one was "nominated for oblivion". No controversy makes sense, thanks to the duality of 2nd order systems, it is normal to have two valid analogies between electricity and any other 2nd order domain.

2.3 Acoustic Analogy

Using gas pressure and velocity as fundamental quantities gives another analogy which helps understanding the acoustic resonators, the model of which is the Helmholz resonator.


Opening Area A
Neck length L
Volume V
Resonant frequency
$f = \frac{v}{2\pi}\sqrt{\frac{A}{VL}}$
where v=velocity of sound

In this resonator :

- elastic energy is stored in the cavity, where the gas is compressed and expanded alternately
- kinetic energy is stored in the neck pipe, where the velocity of the gas is significant

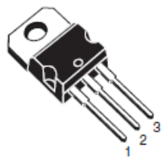
Other acoustic resonators exist that where there are no separated components for elastis energy and kinetic energy, these systems are said "distributed". This happens as well in electronics, in high frequency systems (like more than 500 Mhz), resonators can have L, R and C merged in a single device.
Other analogies based on standing waves are helpful in such cases.

Acoustic resonators are important in nature (they are essential to our ability to speak), in arts (music) and in many industrial applications

2.4 Heat conduction analogy

This analogy is simpler, being restricted to the first degree. Nevertheless it is helpful for managing conduction of heat in solids.

Heat flow in solid	*Electrical domain*
Temperature (°C) T	Voltage (V) V
Heat flow (thermal power) (W) P	Current (A) I
Thermal Energy (J) E = P*t	Electric Charge (C) Q = I*t
Thermal resistance (°C/W) R = T/R	Resistor(Ω) R = V/I
Thermal capacity (J/°C) C = E/T	Capacitance (F) C = Q/V = I/(dV/dT)

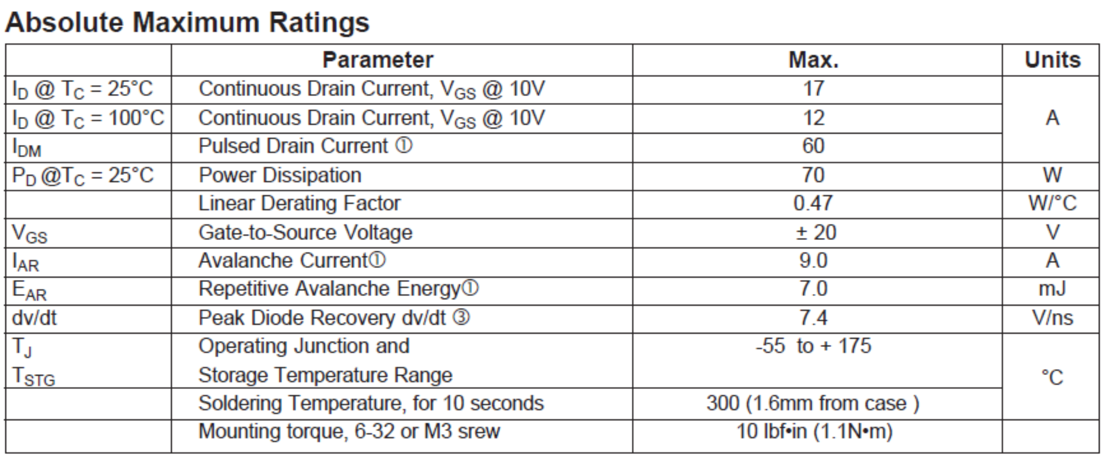


TO-220

This analogy of a great importance for the modeling of thermal behavior of electronic components like transistors, ICs and LEDs.

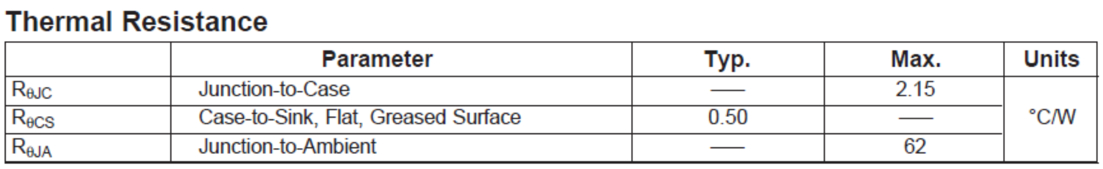
Behaving like RC low-pass filters, the heat dissipators can smooth peaks of thermal power if properly dimensioned.

As an example, the tables below are taken from the datasheet of the IRF530 power transistor in TO-220 package:



Absolute Maximum Ratings

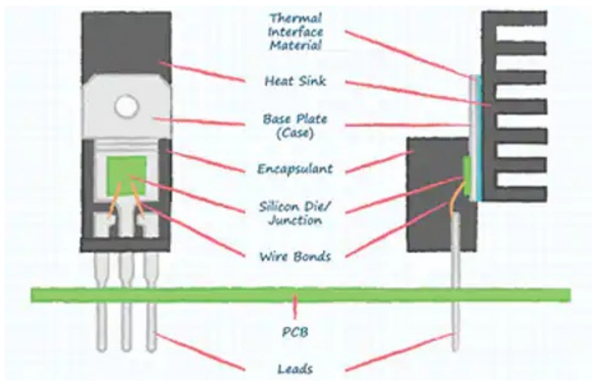
	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	17	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	12	
I_{DM}	Pulsed Drain Current ①	60	
P_D @T_C = 25°C	Power Dissipation	70	W
	Linear Derating Factor	0.47	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
I_{AR}	Avalanche Current①	9.0	A
E_{AR}	Repetitive Avalanche Energy①	7.0	mJ
dv/dt	Peak Diode Recovery dv/dt ③	7.4	V/ns
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to + 175	°C
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 srew	10 lbf•in (1.1N•m)	



Thermal Resistance

	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	——	2.15	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	0.50	——	
$R_{\theta JA}$	Junction-to-Ambient	——	62	

In the maximum ratings, one can read "power dissipation at Tc = 25°C" equals 70 W. Tc is the case temperature, precisely the temperature of the bottom face of the metal tab with the mounting hole.



On the next line one reads "linear derating factor 0.47 W/°C". This means that the power dissipation should be reduced by 0.47 W for each degree of temperature above 25°C. It looks like the relation between the heat flow and the temperature difference between the silicon chip and the case. We observe that :

- the power dissipation should be reduced to zero when the temperature is 25 + (70/0.47) = 173.9°C, which is consistent with the maximum junction temperature of 175°C
- this "linear derating factor" is in fact the thermal conductance between the silicon chip junction and the case tab, the corresponding thermal resistance is 1/0.47 = 2.13 °C/W, which is redundantly indicated as junction-to-case thermal resistance in the bottom part of the table

This transistor is designed to be bolted on a heat sink made of a massive piece of heat conductive material like aluminium or copper. In this case the total thermal resistance will be the sum of the junction-to-case resistance Rjc, the case-to-sink resistance Rcs, and the sink-to-ambient resistance which depends on the shape and dimensions of the heat sink.

The transistor may work without heat sink, in this case its total thermal resistance of 62°C/W determines a maximum continuous power dissipation of (175-25)/62 = 2.4 W only.
Shorts peaks of greater power could be accepted thanks to the thermal capacity of the transistor case (see "pulsed drain current").

Document links :

- en.wikipedia.org/wiki/Voltage
- en.wikipedia.org/wiki/Capacitance
- www.nist.gov/pml/owm/metric-si/si-units
- en.wikipedia.org/wiki/International_System_of_Units
- en.wikipedia.org/wiki/Communicating_vessels
- A-NEW-ANALOGY-BETWEEN-MECHANICAL-AND-ELECTRICAL
- en.wikipedia.org/wiki/Acoustic_resonance

3. Circuit Simulation

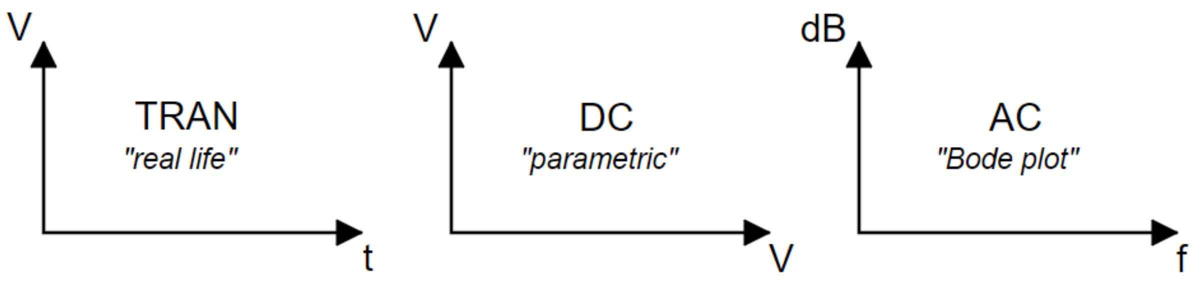
Before the advent of the integrated circuit technology, engineers used to develop their designs by means of prototype which they could fine tune on the laboratory bench.
They were helped by adjustable components (resistors, potentiometers, capacitors, inductors), and replacing a component, moving some connexion or even rebuilding a prototype was affordable.

In the other hand, an integrated circuit cannot be modified and sending a design to manufacturing is a huge money investment, and implies a significant delay. (Laser trim is usable to make fine calibration adjustments for precision integrated circuits but is not suitable for fixing design errors.)

So there was a very strong demand for a simulating solution, and this led to the development of the most successful simulation program in computer history, named SPICE (Simulation Program with Integrated Circuit Emphasis), which was relased by the University of California, Berkeley in 1975, as an open-source project.
Since then, generations of programmers improved the program, building many variants, but the core algorithm remains the same.

3.1 What all SPICE simulators have in common

- Lumped elements circuits (nodes and branches)
- Solution of non-linear equations by means of numerical approximations
- Circuit described in a text files, called netlists, referencing component models also described in text
- Results provided in raw form
- Hierarchical description possible, by means of subcircuits
- A mandatory reference node or "ground"
- Signal sources described as regular circuit elements
- Three fundamental simulation modes :
  - **Transient analysis** : voltages and currents are computed as discrete functions of time, for a finite duration starting from an pre-defined initial state
  - **DC analysis** : voltage and currents are computed for a steady state, time does not appear in calculation. The calculation may be repeated for a finite number of values of one parameter or more.
  - **AC analysis** : frequency domain analysis : complex voltages and currents are computed as discrete functions of frequency. This is known as "small signal analysis" since it requires every non-linear element to be represented by a linear approximation, valid only for "small" signal amplitudes.
- Number formats : suffixes like **f, p, n, u, m, k, meg, g, t** may be used for femto, pico, nano, micro, milli, kilo, mega, giga, tera. They must be appended to the number with no space.
  Warning: a frequent user error is to use M for mega, while the simulator understands it as milli.
- The user needs to be aware of stability and convergence issues



3.1.1 Transient analysis

Before starting the transient simulation, the program automatically runs a DC analysis in order to determine the initial value of each voltage and current.
It happens sometimes that this fails, if the circuit is unstable (like an ascillator), or has too many possible steady states (like a digital sequential system). In these cases, the initial state determination must be helped, explicitely or by means of the signal sources.

The only mandatory parameter the user has to specify is the duration or "stop time".

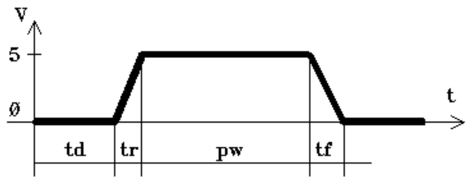
Another important parameter is the "maximum timestep".
For efficiency reasons, the timestep or interval between successive computed instants is dynamically adjusted by the program.
But it may happen that "smoother" results are desired, in this case a maximum timestep may be specified, at the expense of more computation time.
Conversely, it may happen that the automatic timestep approaches zero and the simulation stalls. In this case the circuit needs to be fixed.

3.1.2 Signal sources for transient analysis

IMPORTANT : no instantaneous variation of signal (ideal step function) should be specified, this does not exist in the real world and would prevent the algorithm convergence.

There are several ways to specify a signal source (either voltage or current) as a function of time for transient analysis, two of these deserve a special attention:

- The **pulse source**, which creates a periodic pulse defined by two voltages and 5 relative time values
  td, tr, tf, pw and period
  .

  A source of frequent errors with the pulse is the fact that the pulse duration **pw** is specified after the fall time **tf**, which is counter-intuitive
- The **piecewise linear source (PWL)**, which is defined by an indefinite list of couples of signal value and absolute time value
  The PWL source is much more general, but requires more parameters than the pulse source in simple cases.

3.1.3 DC analysis

A frequent use of the DC analysis is making characteristic curves or static transfer functions similar to those printed in components datasheets. For this purpose, the simulator permits to sweep the value of a signal source, which in fact is repeating the DC analysis for a finite number of values of the chosen source. This is specified in the analysis directive, overriding the source description.

In addition, it is possible to request a repetition of this sweep for a limited set of values of a second source or parameter.

3.1.4 AC analysis

The main use of the AC analysis is the production of "Bode plots", which are curves with :

- on the horizontal axis, the frequency of the signal, with a logarithmic scale
- on the vertical axis, the relative magnitude of the signal expressed in dB (which is another kind of logarithmic scale), and the phase of the signal in degrees

For this purpose, the user needs to specify an AC value and an AC phase for each source in the circuit, in addition to the DC value which is used to compute the steady state of the circuit. In simple cases, an AC value of 1 and a phase of 0 are convenient.
Note: the AC value is an abstract reference, it does not need to be "small" and it does not interfere with the fact that we are doing a "small signal analysis".

3.1.5 Digital Circuits

It is possible to simulate digital circuits with SPICE, but to check a logic design it is more efficient to calculate only a boolean value for each node rather than a voltage, so many digital or logic simulators have been developed.
For this purpose, two hardware description languages have been created, named VHDL and Verilog (one would have been sufficient, there are two for historical reasons).
These languages are also used to drive logic optimizers, in the context of FPGA and digital integrated circuits design.

3.2 LTSPICE in practice

LTSPICE is a modern SPICE simulator which was created by a semiconductor company called Linear Technology, later acquired by a company called Analog Devices. So nowadays (in 2023) LTSPICE is freely offered by Analog Devices

LTSPICE offers in a single application program:

- a simple schematic editor, with hierarchical capability
- an improved SPICE simulation engine, with added behavioral modeling capabilities
- a graphic waveform viewer

3.2.1 The schematic editor

The operation of the editor is pretty obvious.
Notice that the right mouse button allows to abort the current command and otherwise gives access to a context menu.
There is no "Edit" menu with "copy, paste, delete", instead in the toolbar there is a copy button (which, in fact, duplicates) and a delete button (scissors)

Components libraries

The toolbar has buttons for some frequently used components like ground, resistors, capacitors, etc... , then every other component can be found in the components libraries (the button representing a logic AND gate, quite misleading).
Note that the independant voltage source (providing DC, pulse, PWM, etc...) is simply named "voltage".

Labels

In order to be able to find the desired nodes for plotting, it is recommended to put a label on each one (the button representing a sign).
Note that unconnected wires bearing the same label are simulated as a single net, this may be used to simplify the drawing of common signals like power supplies.

Simulation commands and other SPICE directives

The simulation commands are stored on the schematic in text form, using the legacy SPICE syntax.
They can be generated easily using the Edit Simulation Command in the Simulate menu.

Other SPICE directives, known as "dot commands" because they begin with a '.' character, can be placed using the last button ".op" of the toolbar.

A notable example is the **.step** directive which can be used to iterate a simulation (of any type) for different values of a component parameter

Parametrized design

An arbitrary parameter can be created with the .param SPICE directive, to be used in a parametric expression:

```
.param R1=44k
.step param R1 1k 5k 2k
```

In the example above, the simulation is iterated with parameter R1 swept from 1k to 5k with step 2k, which results in simulating with 3 values, 1k, 3k, 5k.

In a component instance, any numeric value can be replaced by a parametric expression involving constants and parameters. The expression should be enclosed in curly braces **{}** (even if it contains only a single parameter)

```
{R1*(k+1)}
```

Hierarchical design

To create a cell or subcircuit to place into a hierarchical schematic, do the following :

1. create the schematic of the cell as a regular schematic
2. define each terminal of the cell by placing a label on the relevant node, with Port Type set as Input, Ouput or Bi-Direct
3. after saving the schematic, select Open this Sheet's Symbol in the hierarchy menu, the program will propose to generate a symbol, accept
4. if desired, modify the outline of the symbol and the placement of the pins

Then the new symbol is available to be inserted in a higher level schematic, it can be found in the component library, after changing the top directory to the relevant location.

A subcircuit containing one parameter (or more) can receive a parameter value from the higher level schematic, by adding a parameter value or expression to the symbol instance:

```
W=3.33
K={2.0*N}
```

In the example above, W and K are parameters belonging to the subcircuit, while N is a higher level schematic parameter

3.2.2 Behavioral modeling

The component named **bv** is a configurable voltage source, the value of which can be defined as an arbitrary function of time, node voltages, device currents, including time derivative and time integral of these voltages and currents. Likewise, the component named **bi** is a configurable current source.

Note: the simulator is able to remove any discontinuity of the model by smoothing the function, but it is advisable to favor continuous functions.

3.2.3 Waveform viewer

When a simulation ends, the waveform viewer opens an empty black window. Then signals can be added usind Add Trace in the context menu.

Signals are overlayed in the same horizontal strip. To view them separately, create more strips with Add Plot Pane

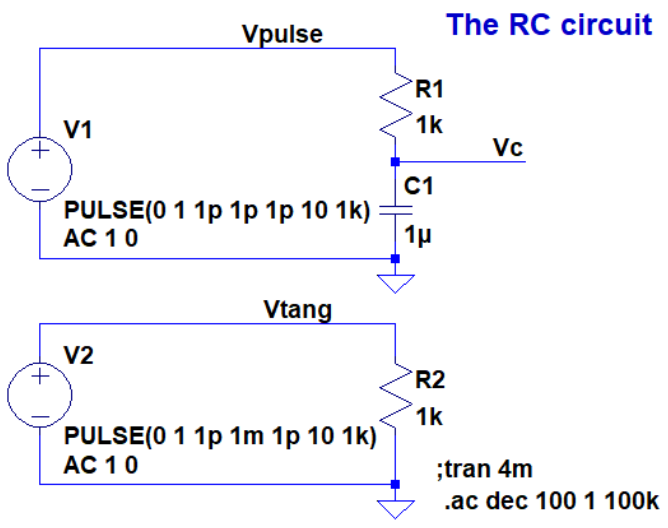
A cursor can be added by clicking on a signal name, then it can be dragged horizontally. A second cursor can be added by clicking again on the same name.

3.3 Simulation examples

3.3.1 Transient simulation example

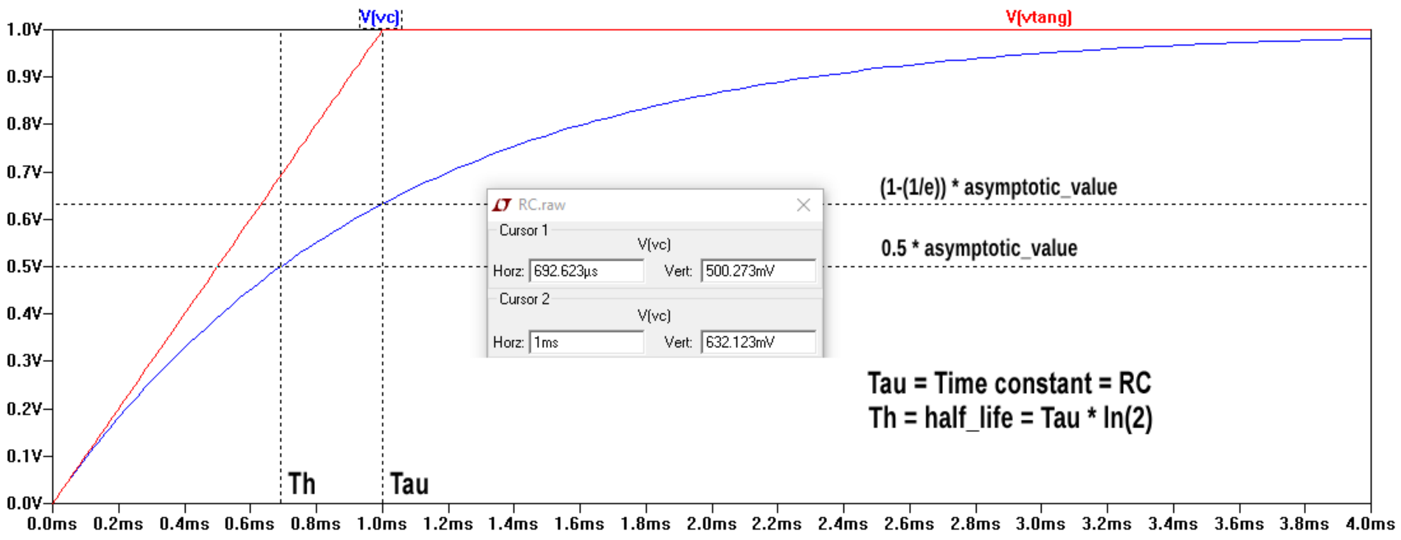
In this example, we consider the RC circuit, which is a classical illustration of a system governed by a differential equation of the first order.
The circuit is made of R1 and C1, it is operated with a voltage step of 1 Volt generated by V1, its output is labelled as Vc.

For illustration, another voltage source V2 generates a voltage Vtang, the plot of which is supposed to be tangent to Vc, according to the theory.



LTSPICE source file

Plot of the voltage Vc across C1 :



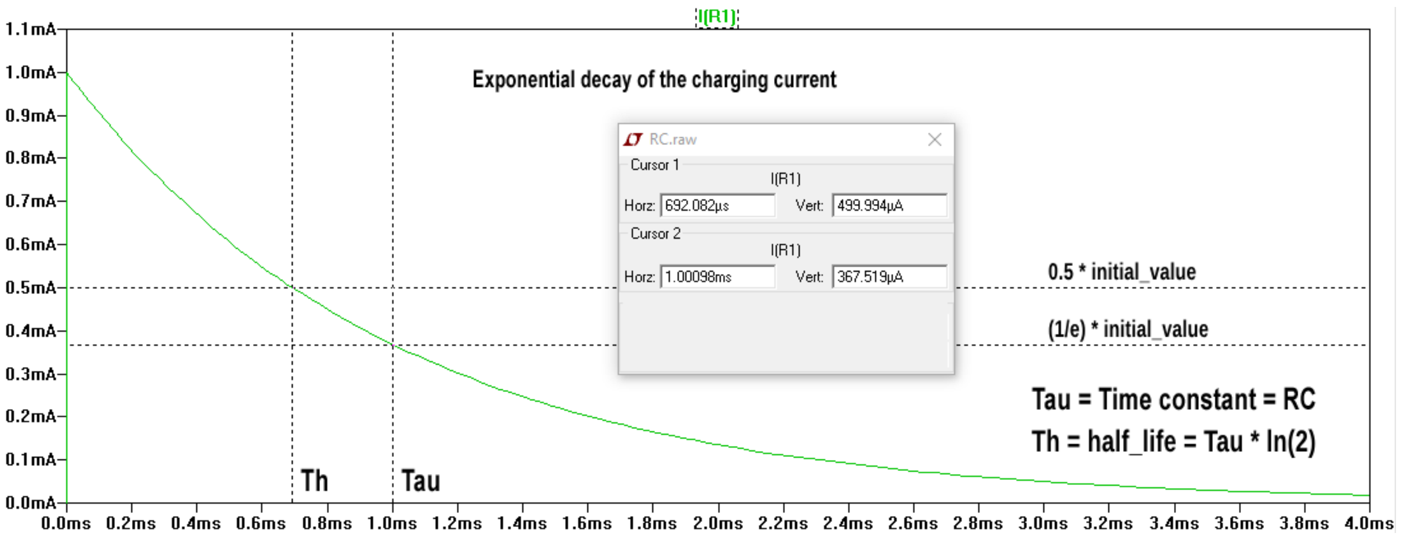
```
Cursor 1
                V(vc)
Horz: 692.623µs     Vert: 500.273mV
Cursor 2
                V(vc)
Horz: 1ms           Vert: 632.123mV
```

We observe a function Vc(t) = Vstep * (1 - exp(-t/Tau)). The **time constant** Tau is a characteristic parameter of the circuit

- Tau = R * C (R being the effective resistance viewed from C)
- Tau is the inverse of the slope of the curve at the start point
- Tau is the time it takes to reach the asymptotic value multiplied par 1 - 1/e (approx. 0.632)
- Tau is related to Th, the time it takes to reach half the asymptotic value :
  Th = Tau * ln(2) (approx. Tau * 0.693)

Plot of the current through the resistor:

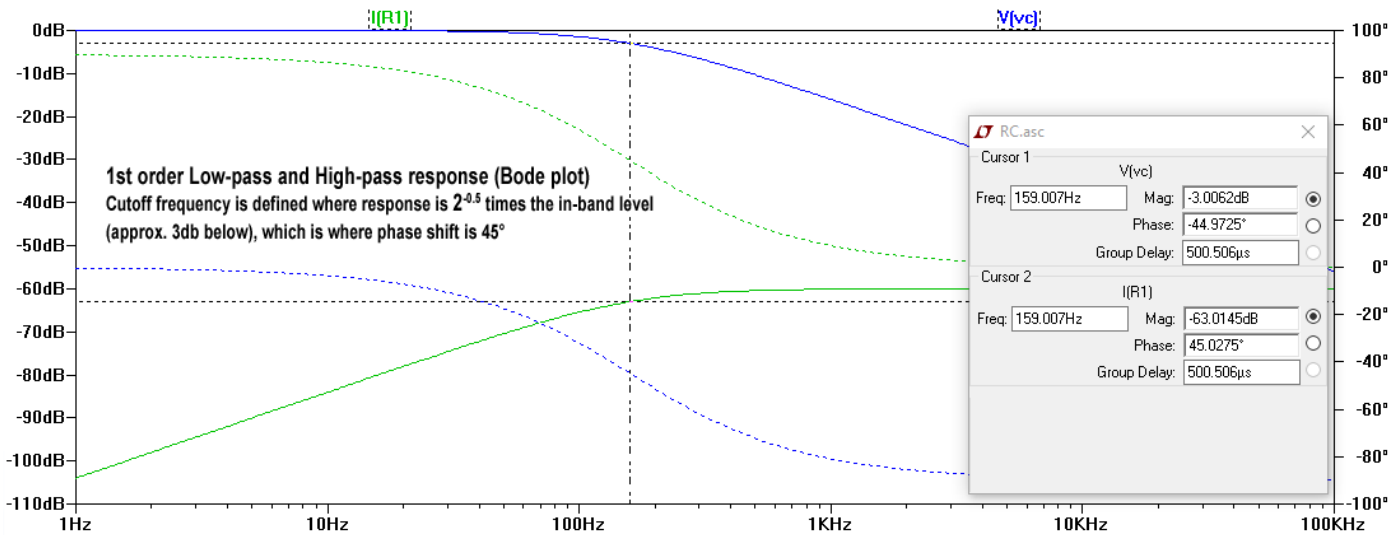
(or the voltage across the resistor which is proportional)



```
Cursor 1
                I(R1)
Horz: 692.082µs     Vert: 499.994µA
Cursor 2
                I(R1)
Horz: 1.00098ms     Vert: 367.519µA
```

3.3.2 AC simulation example

Based on the same example, the AC simulation produces the Bode plot of Vc and I(R1).
The simulation command .ac dec 100 1 100k specifies 100 points per decade, from 1Hz up to 100kHz.



1st order Low-pass and High-pass response (Bode plot)
Cutoff frequency is defined where response is $2^{-0.5}$ times the in-band level
(approx. 3db below), which is where phase shift is 45°

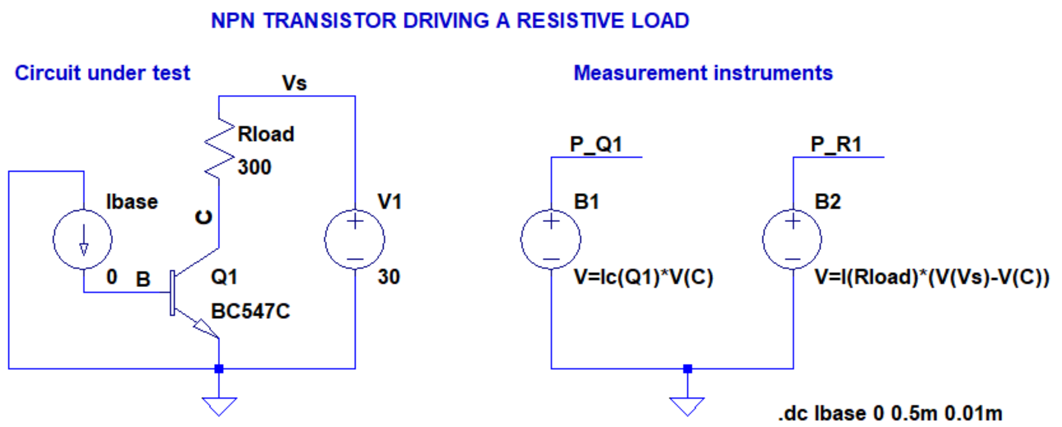
Cursor 1
V(vc)
Freq: 159.007Hz Mag: -3.0062dB
Phase: -44.9725°
Group Delay: 500.506µs

Cursor 2
I(R1)
Freq: 159.007Hz Mag: -63.0145dB
Phase: 45.0275°
Group Delay: 500.506µs

Considering Vc as the output, this circuit is the simplest low-pass filter (first order).
The cutoff frequency (in rad/s) is 1/Tau.
Likewise, I(R1) expose a high-pass filter function with the same cutoff frequency.
By permutation of R1 and C1, a high pass filter with a voltage output could be obtained.

3.3.3 DC simulation example

This example demonstrates a bipolar junction transistor (BJT), NPN type, used to vary the power applied to a resistive load.
The typical current gain of this transistor is 200. The transistor base current is driven by the simulation directive, from 0 to 0.5mA, which is expected to be sufficient to cause the load current to reach 100mA.

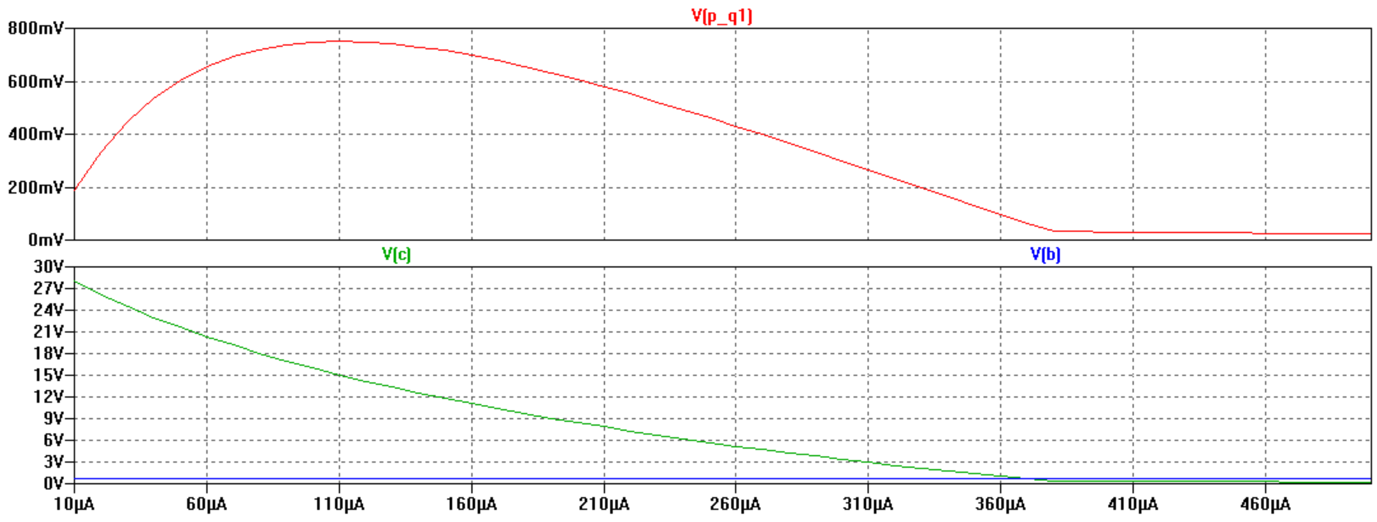


```
.dc Ibase 0 0.5m 0.01m
```

LTSPICE source file
BC547 transistor datasheet

Two behavioral voltage sources are added in order to compute the power dissipated in the transistor and in the load.
Note: the voltages at nodes P_Q1 and P_R1 represent watts of power.

We observe that the maximum power dissipated in the transistor exceeds the static power limit of this transistor (625mW), so this circuit is suitable for switching the load on and off but not for varying its power continuously.

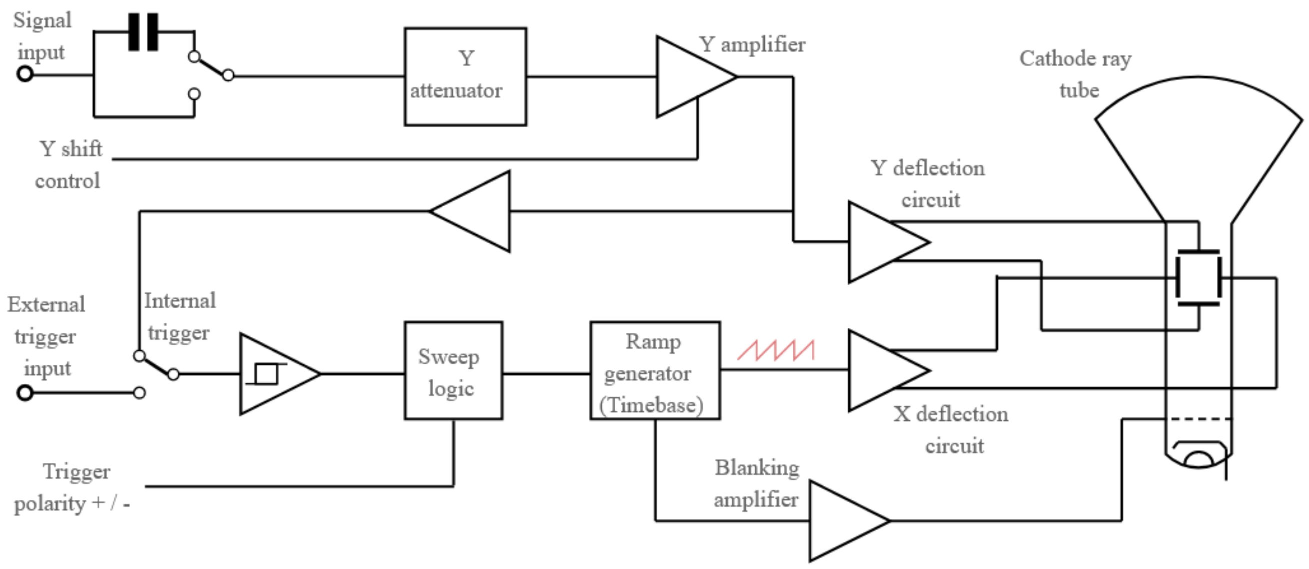


Theory links :

- en.wikipedia.org/wiki/RC_circuit
- en.wikipedia.org/wiki/RC_time_constant
- en.wikipedia.org/wiki/Bode_plot
- en.wikipedia.org/wiki/Bipolar_junction_transistor

4. The Oscilloscope

The oscilloscope is an measurement instrument, the purpose of which is to display a signal as a function of time.
Before it became available, such displays could be obtained by mechanical or electro-mechanical instruments like the analog pen plotter and the oscillograph.
The invention of the cathode ray tube (CRT) allowed to build an electronic instrument with a much wider bandwidth : the oscilloscope.

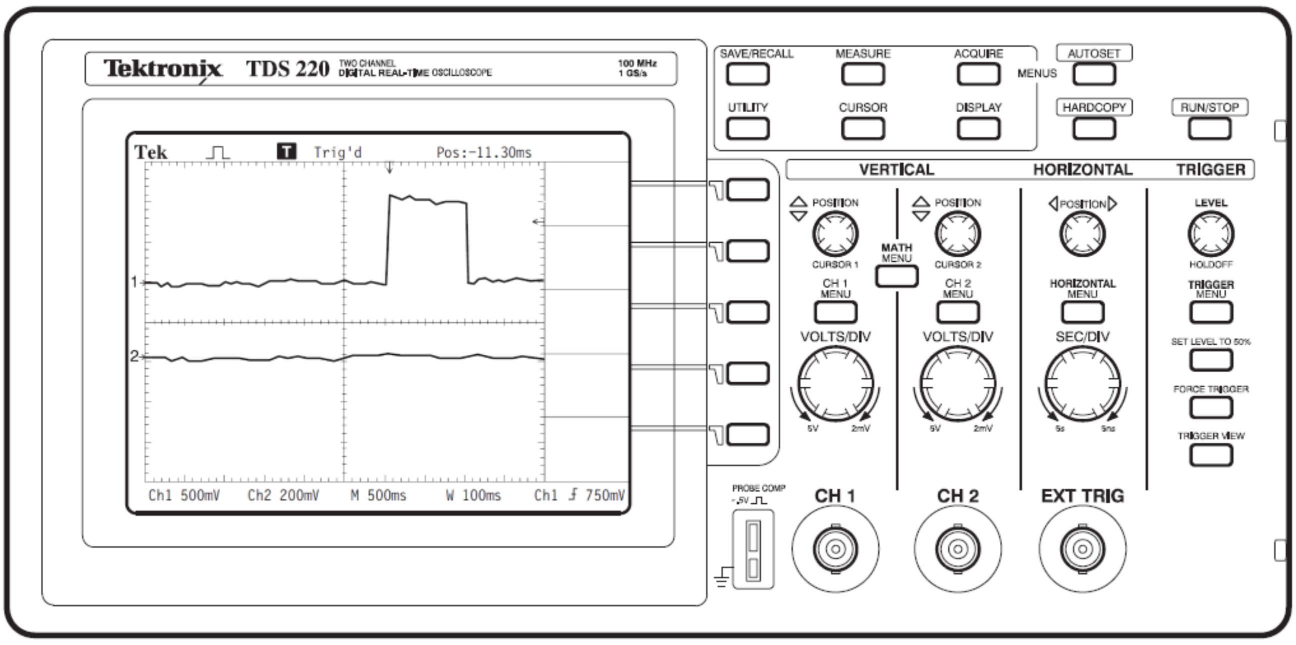


- the vertical axis "Y" displays the value of the amplified input voltage, with a precise scale (Volts/division)
- the horizontal axis "X", being driven by a ramp signal, represents the time with a precise scale (seconds/division) the periodic ramp signal must be synchonized with the input signal in order to maintain a readable image

Basic CRT oscilloscope could display only periodic signal due to their lack of memory.
Storage oscilloscope, with an electrostatic memory built inside the CRT, were able to store a single sweep for examination, but these were very expensive.

Digital storage oscilloscope (DSOs) convert, store, process and display the signal in digital form, offering a similar service at a much lower price.
However the basic concepts have not changed since the analog era, and the functional diagram reproduces the same signal paths.

4.1 The standard benchtop oscilloscope



Legacy analog oscilloscope had one knob for each parameter and one push button for each option - this was very user-friendly but expensive and bulky.
Modern instrument have knobs for the most frequently used parameters, and buttons for selecting on-screen menus for less frequent adjustments. Pocket instruments may have as little as one single knob and four buttons.

In the standard layout controls are normally distributed in 4 zones:

- Vertical : input channels
- Horizontal : time base
- Trigger : synchronization of the horizontal sweeping
- Utilities : display options, measurements, cursors

4.1.1 Vertical controls

There is one vertical control panel for each input channel, in the example above we have a 2-channel oscilloscope. Each channel has a coaxial input connector, the most common type is the BNC connector.

Two main knobs are always present : the vertical scale (Volts/div) and the vertical position. In spite of being digitally handled, these buttons control analog circuits : the gain of the input amplifier and the offset voltage which is added to the signal before the analog-to-digital conversion (ADC).

Other controls are possibly carried over an on-screen menu :

- Channel On/Off
- Coupling (DC/AC) switch : DC is the "normal" option. The AC position inserts a capacitor in the input signal path, in order to remove the DC component. This is very useful when people need to observe small variations (AC) of a signal having a large quiescent value (DC)
- X1/X10 switch : this switch allows to make the displayed value of the vertical scale consistent with the type of probe in use. It has no effect on the signal path.
- Low pass filter : used to remove high frequency noise

4.1.2 Horizontal controls

Two main knobs are always present : the horizontal scale (seconds/div) and the horizontal position.

The horizontal scale controls the sampling period, which is the time interval between consecutive analog-to-digital conversions (this is derived from the number of samples per division, which is not usually accessible to the user).

The horizontal position controls the position of the trigger instant on the horizontal space, this instant is made visible by an arrow normally placed on the top edge of the graphics window area, unless it is outside this window.
Note: if the sample memory contents were displayed simply from the beginning to the end, the trigger instant would be stuck to the left edge of the graphics like in legacy oscilloscopes. A special memory management scheme called "circular buffer" allows to move this instant, actually displaying samples which were acquired before the trigger event.

4.1.3 Trigger controls

The main knob is the level control, which determines the threshold voltage. A trigger event is generated when the signal crosses this threshold in a determined direction (rising or falling). This level is made visible by an arrow normally placed on the right edge of the graphics window area, unless it is outside this window.

Other controls are possibly carried over an on-screen menu :

- Source : one channel may be chosen as a source for the trigger signal, alternately an auxiliary signal input may be used as an external trigger source
- Slope or Edge: the rising edge or the falling edge
- Sweeping mode :
  - Normal : each time the sweeping reaches the end of the window, the process is paused, waiting for the next trigger event.
    Note: if the signal does not cross the trigger threshold, the display remains frozen.
  - Auto : After a limited waiting time, the sweeping is forced to restart instead of remaining frozen, in order to let the user keep an eye on the signal while adjusting the trigger parameters
  - Single : after a single triggered sweep, the display remains frozen until the user manually restarts the process, by pushing the Run/Stop button

The Run/Stop button is used to pause the sweeping, or to restart it, notably in the case of the single sweep mode.

4.1.4 Utilities

These are miscellaneous functions which may include :

- Measurements : the instrument displays measurements computed from the contents of the sample memory, like peak values, average value, RMS value, peak-to-peak amplitude, estimated frequency, etc...
- Cursors or Markers : the instrument lets the user draw cursors on the graphics and read their positions and distances
- Signal processing functions, like averaging consecutive sweeps in order to remove noise, or computing a spectrum using Fourier Transform (FFT)
- Display preferences

4.2 The PC-based oscilloscope

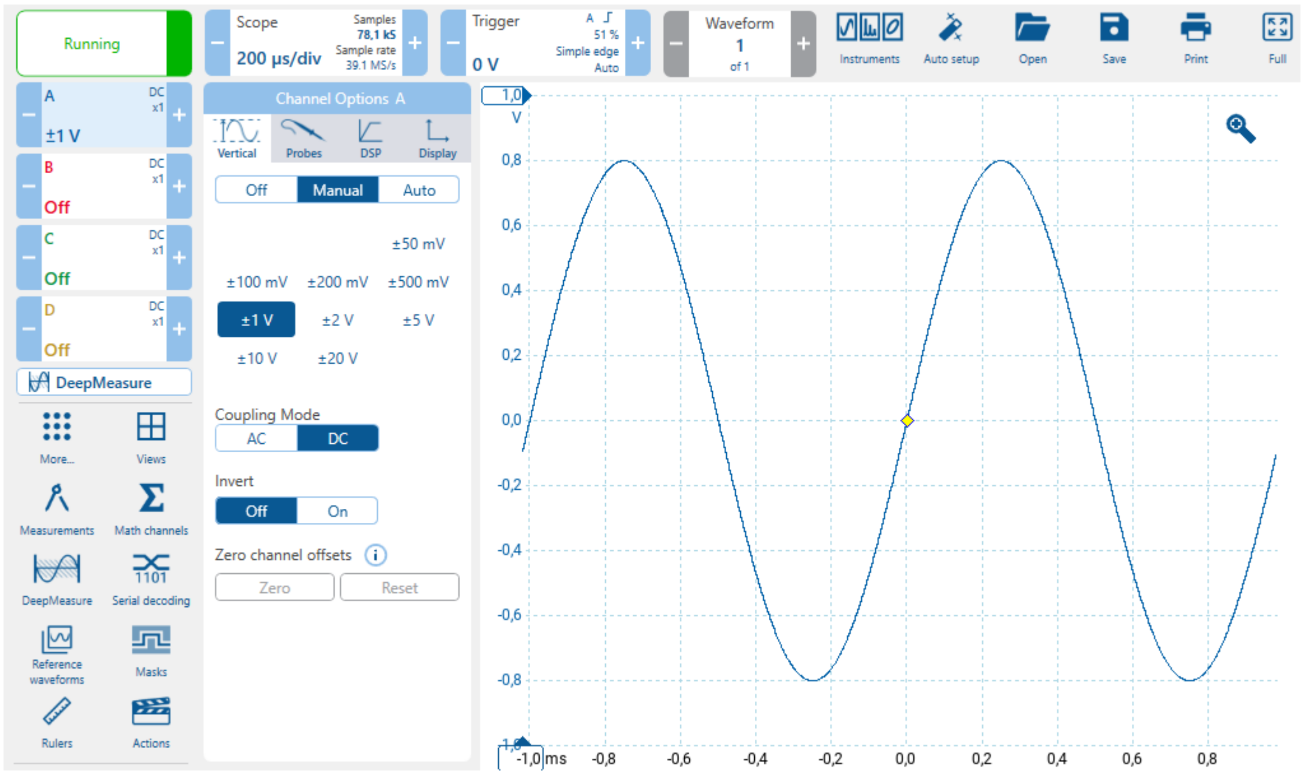
Another approch permitted by the digital technology is to make a "black box" containing the specific hardware like input amplifiers with gain control, ADCs, sample memory, trigger detectors, time base, and to connect this to a standard computer where the user interface and graphics display will take place.

4.2.1 Pros and Cons

- Benefits :
  - lower cost
  - possible extension of graphics capabilities, resolution, colors etc...
  - easier post-processing and archiving of results, including custom software development
  - possible inclusion of other instruments in the same black box, example: a signal generator
- Disadvantages:
  - no knob, all parameters in menus, this ends up requiring more actions from the user for a given task in a professional context
  - more sophisticated user interface wich may be confusing for the new user
  - screen updated with a noticeable latency (delay)
  - occupation of more space on the laboratory bench

4.2.2 An example : the Picoscope

The Picoscope is a black box with coaxial BNC inputs for the signals, and a USB connexion to the computer. It comes with a specific application software, which presents the 4 control zones of a standard digital oscilloscope described above.



Running

Scope 200 µs/div — Samples 78,1 kS — Sample rate 39.1 MS/s

Trigger 0 V — A 51 % Simple edge Auto

Waveform 1 of 1

A DC x1 ±1 V

B DC x1 Off

C DC x1 Off

D DC x1 Off

Channel Options A

Off [✓ Manual] Auto

±50 mV ±100 mV ±200 mV ±500 mV [✓ ±1 V] ±2 V ±5 V ±10 V ±20 V

Coupling Mode: AC [✓ DC]

Invert: [✓ Off] On

Zero channel offsets

On the screen image above, one can see :

- Run/Stop button : the green button on the top left corner, which turns red when the acquisition is paused
- Vertical : along the left border, the channel buttons, labelled from A to D
  pressing one of these buttons causes the corresponding channel menu to be displayed,
  the '-' and '+' allow a quicker change of the vertical scale setting
- Horizontal : on the top border, the "Scope" button giving access to the timebase menu,
  the '-' and '+' allow a quicker change of the seconds/div setting
- Trigger : on the top border, the "Trigger" button giving access to the trigger menu,
  the '-' and '+' allow a quicker change of the trigger level setting
- Utilities : among many utilities, the Measurements and the Rulers (cursors) give services similar to their counterpart on the benchtop oscilloscope

The user must be aware of some disturbing differences with respect to the standard oscilloscope:

- the vertical scale is represented as a full range (exemple +-50mV) instead of volts/div
- there is no vertical position control, it is replaced by an offset setting in the Display tab of the vertical menu, it is expressed in % instead of volts, its range is restricted to +-90%
- the trigger level and trigger horizontal position are represented together by a diamond shaped symbol in the middle of the graphics instead of the arrows along the sides
- the trigger level is restricted to the vertical range
- the trigger instant is expressed in % instead of seconds and is restricted to acquisition time span

IMPORTANT WARNING : PC based oscilloscope offer some king of zooming on the displayed graphics. This should not be confused with actions on the vertical and horizontal acquisition scales:

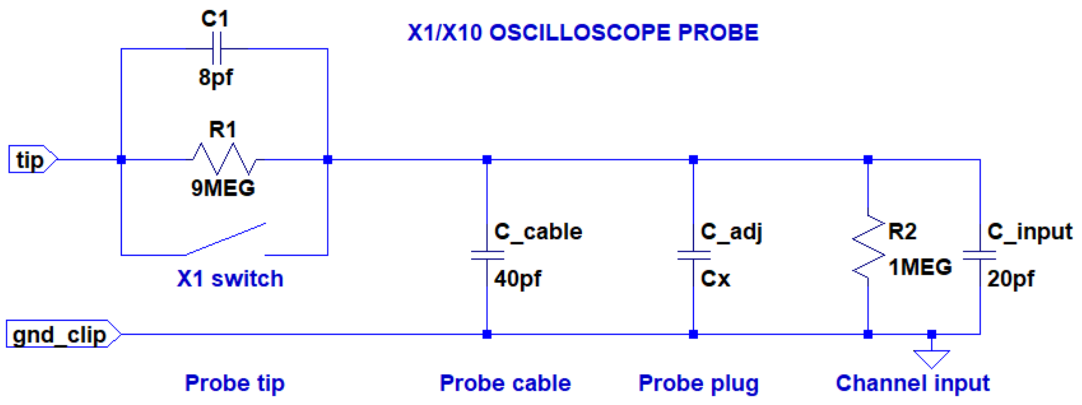
- Zooming changes only the way stored data are displayed
- Changing the acquisition scales is taken into account at the next acquisition, the resolution of which is then modified, possibly offering more detail

4.3 The oscilloscope probe

A measurement instrument should have a minimal impact on the system under test, for this reason the input current of the oscilloscope probe should be minimized. A widespread solution is the "X10" probe :

The probe attenuates the signal by a factor of 10, so the signal is actually multiplied by 0.1, while the Volts/Div setting is multiplied by 10, whence the "X10" label.

The attenuation being done by a voltage divider, the input resistance is multiplied by 10, resulting in an input current divided by 10.



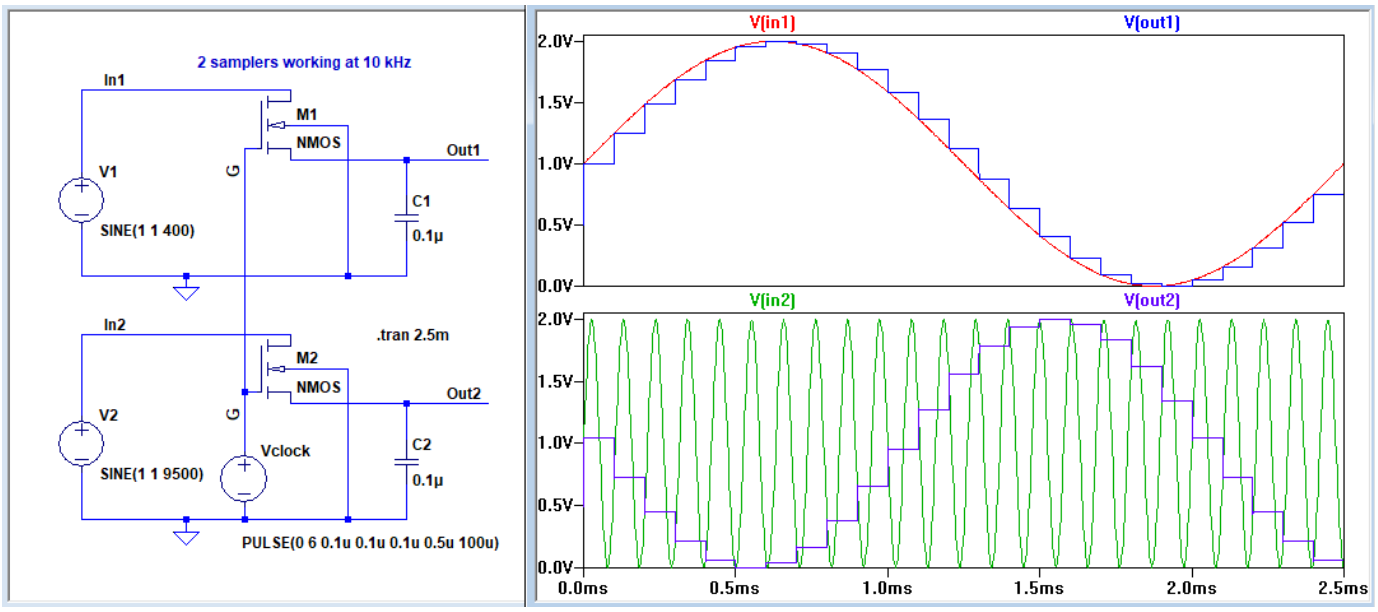
The oscilloscope channel exposes a standard input resistance of 1 MOhm (1000000 Ohms), so putting a 9 MOhms resistor in series makes a voltage divider, dividing the signal by 10.
This attenuation may be disabled by a switch short-circuiting this resistor.

In high frequencies, the capacitances interfere with the signal attenuation ratio, so the C1 capacitor is inserted in the probe tip in order to make a capacitive voltage divider together with C2 = C_cable + C_adj + C_input.
C_adj is an adjustable capacitor allowing to precisely match the capacitive attenuation with the resistive attenuation of 1/10.

4.4 The aliasing issue

Digital instruments periodically sample the signal. The sampling theory (Nyquist–Shannon theorem) states that the sampling frequency should be greater than twice the signal frequency.

If the original signal frequency exceeds half the sampling frequency, an alteration known as "aliasing" causes the sampled signal to expose a wrong frequency. This is also known as "stroboscopic effect" or "wagon-whell effect".



In the example above, an input signal is sampled by means of a MOS transistor and a capacitor. The MOS transistor works like a switch, which is driven by a periodic signal composed of short pulses at 10 KHz. Each sampler keeps the sampled value constant for an entire sampling period to permit the ADC operation. This creates the typical staircase aspect.

Two samplers are simulated simultaneously for comparison purpose.

- The top sampler receives a 400 Hz signal at its input, its output presents an acceptable sampled representation of the input signal.
- The bottom sampler receives a 9500 Hz signal, but its output presents a wrong frequency of 500 Hz only, which is called an "alias" of the original signal. This happens because the input frequency is greater than half the sampling frequency.

If a periodic signal is observed with a seconds/div setting which is thousands times greater than the signal period, the digital oscilloscope may display the signal with an totally wrong apparent frequency, with no kind of warning.

If the signal frequency is totally unknown, it is advisable to start observing it with the smallest seconds/div setting and then increasing it until obtaining a convenient display.

LTSPICE source file for the illustration

Theory links :

- the Oscilloscope Course by Tyco
- en.wikipedia.org/wiki/Circular_buffer
- en.wikipedia.org/wiki/Stroboscopic_effect
- en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

5. The Microcontroller

A microcontroller is a computer on a chip, designed to work with a minimal number of external components - in fact some can work with no external component at all.

Microcontrollers are manufactured to be embedded in products which are no perceived as computers by the end-user. Due to their low cost and the flexibility of software solutions, they make an increasing fraction of the electronic hardware in general.

Like every computer, a microcontroller is a digital system, made of logic gates and memories, themselves made of transistors.
While desktop computers may also contain electro-magnetic devices like hard discs and electro-optical devices like CD/DVD units, microcontrollers are just highly complex fabrics of transistors, containing millions of them.

Having no operating system, a microcontroller executes constantly the only program its memory contains, however it is able to execute many tasks concurrently (in parallel).

5.1 Binary words and memory addresses

5.1.1 Words

Internally, computer work on binary data or bits.
Since working on one bit at a time would be inefficient, data are organized in binary words, which are fixed size collections of bits. For interoperability reasons, the word size is frequently a multiple of 8 bits.

An 8-bit binary word is an octet or byte.

5.1.2 Numbers

A binary word may represent a number, for this purpose a weight is assigned to each bit:

- the bit with the smallest weight (which is 1), is the LSB (Least Significant Bit)
- the bit with the greatest weight is the MSB (Most Significant Bit)

Signed integer number representation in computers normally uses the two's complement scheme, while fractionnal numbers said floating point numbers are represented by words divided in three fields: sign bit, exponent and mantissa.

Reading binary data is uneasy for humans, so developers prefer to use base-16 or hexadecimal numbers to represent binary words comfortably in documentation and programs.

5.1.3 Text

A binary word may be used to represent a text character. It appears that 8 bits are sufficient for representing the letters used by latin alphabets, so the starting point was a code known as ASCII, representing all the characters of an american typewriter (for which 7 bits would have been sufficient).
Several solutions were proposed to extend this code to non-latin languages, the most successful being UTF-8, which uses a variable-length code, compatible with ASCII.

A piece of text is represented as a sequence of bytes, called a string. It is a common practice to use the zero byte (null character) to mark the end of the string.

5.1.4 Addresses

In computer memory, data words are stored in 1D-arrays, which can be considered as tables where each line contains a binary word.
The lines are numbered in sequence, and their numbers are called the addresses, which are themselves handled in binary form.

Many computers can read or write words which are larger than 8 bits, but to preserve the capability or writing or reading single bytes, the adresses are always byte addresses.

5.1.5 Instructions

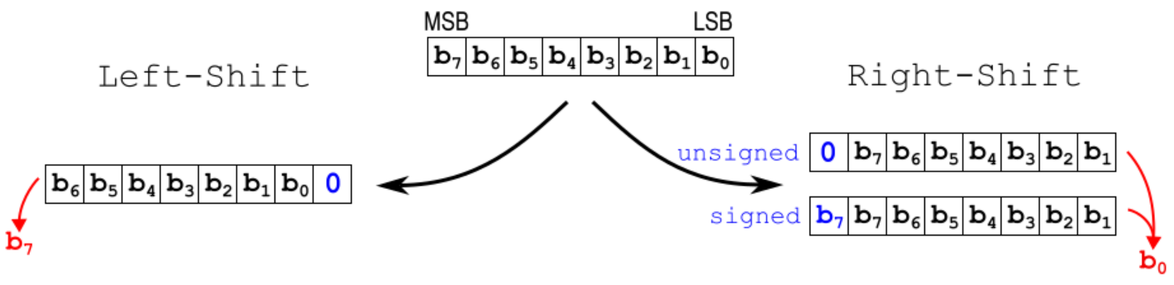
Programs in machine code are sequences of instructions coded and stored as compact binary words. The code is architecture-specific and not readable by humans.
When the instruction word size is fixed, the archicture is called RISC (Reduced Instruction Set Computer), otherwise it is called CISC (Complex Instruction Set Computer).

- Programs written in languages like assembly, C, C++, Java are translated into binary code in advance, by a program called compiler (or assembler).
- Programs written in scripting languages like bash, javascript, python, php are translated during execution, by a program called interpreter.

5.1.6 Bit Level Operations

Bit level operations are implemented on every computer, but they are of a higher importance for the microcontroller developer, since they are involved in nearly every interaction with peripherals.

Bit shift


MSB
LSB
b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
Left-Shift
Right-Shift
b_6 b_5 b_4 b_3 b_2 b_1 b_0 0
b_7
unsigned
0 b_7 b_6 b_5 b_4 b_3 b_2 b_1
signed
b_7 b_7 b_6 b_5 b_4 b_3 b_2 b_1
b_0

Bit shift operations shift bits to the left (which is towards higher weights) or to the right (which is towards lower weights), and there are two modes of right-shifting

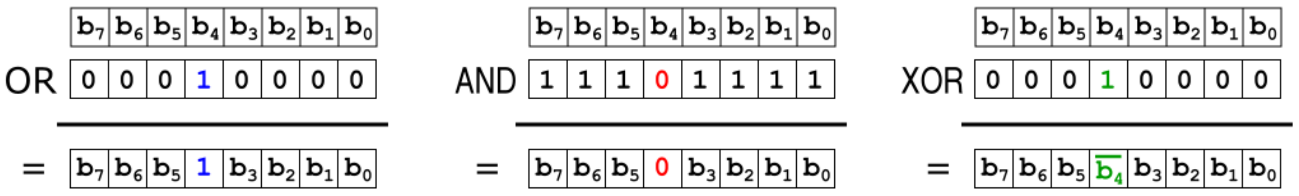
When bits are shifted left by one position, the MSB (Most Significant Bit, b_7 in the example) is dropped, while a zero is injected in the LSB position.
If the word represents a number (not too great), it is multiplied by 2

When bits are shifted right by one position in "unsigned" or "logic" mode, the LSB (Least Significant Bit, b_0) is dropped, while a zero is injected in the MSB position.
If the word represents an unsigned number, it is divided by 2.

When bits are shifted right by one position in "signed" or "arithmetic" mode, the MSB is duplicated, with the intention of preserving the sign of a signed number coded in two's complement.
If the word represents a signed number, it is divided by 2.

In C and C++, these operations are supported by means of the "<<" and ">>" operators. The mode (unsigned or signed) is determined according to the type of the first operand.

Bit set and reset



In the first example, a bit is "forced" to 1 or "set", without affecting the other bits, by means of a bitwise logic OR operation with a constant containing zeros except in one place.

In the second example, a bit is "forced" to 0 or "reset", without affecting the other bits, by means of a bitwise logic AND operation with a constant containing ones except in one place. Such a constant is called a "mask".

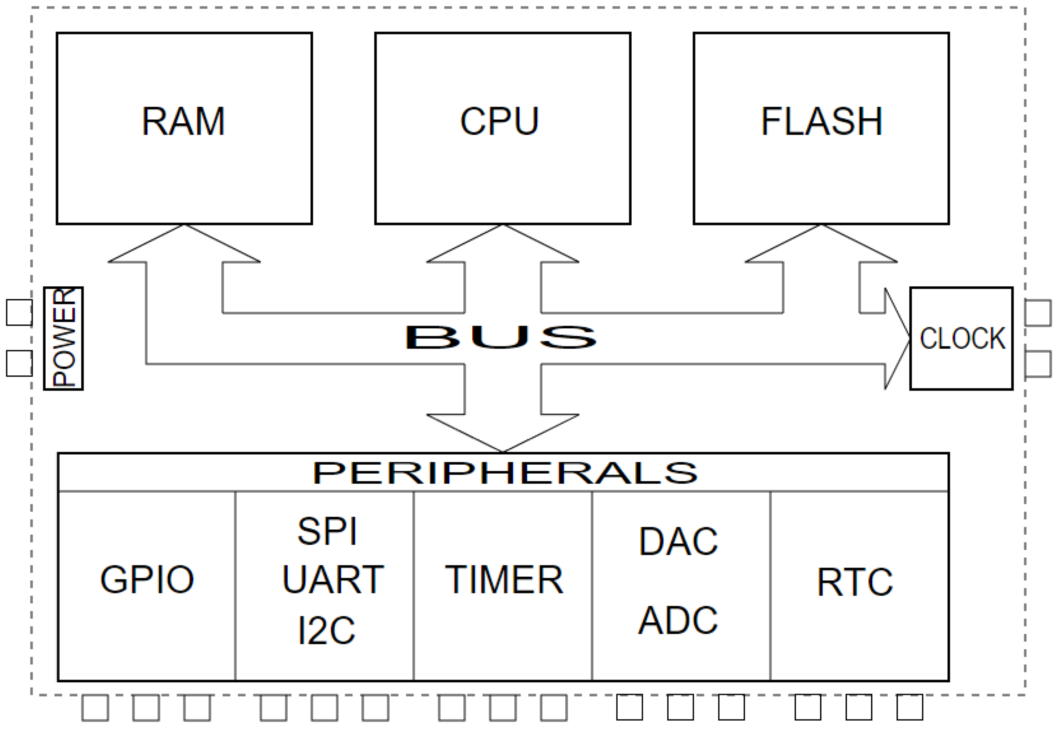
In the third example, a bit is inverted or complemented, without affecting the other bits, by means of a bitwise logic XOR (eXclusive Or) operation with a constant containing zeroes except in one place.

In C and C++, these operations are supported by means of the bitwise logic operators "|", "&" and "^" operators. In addition, the "~" operator performs the NOT function, complementing all the bits.
Note: these should not be confused with the boolean operators "||", "&&" and "!" which handle words considered each one as a single boolean value.

Examples in C :

```
A |=  ( 1 << 5 );  // bit 5 (weight 32) in A is forced to 1
B &= ~( 1 << N );  // bit N (weight 2^N) in B is forced to zero
```

5.2 Microcontroller architecture



The main modules inside a microcontroller are :

- the CPU (Central Processing Unit),
  which processes binary words that it reads and writes from memory and peripheral through the bus
- the non-volatile memory, sometimes called ROM or flash memory or program memory,
  which contains programs and constant data
- the volatile memory, generally called RAM or data memory, which contains the temporary data
- the peripherals, which permit communication with the outer world in various forms
- the bus lines which support the communication between these modules

The parameters that characterize a microcontroller type are:

- the CPU family, which determines the instructions encoding and the performance level
- the word size, either 8 bits, 16 bits, 32 bits or 64 bits,
  this is the typical data word that the CPU can process at once, but the definition of this parameter is not uniform, for example an 8-bit microcontrollers may use 16-bit addresses and 12-bit instructions, but performs calculations on 8-bit data only
- the RAM size
- the program memory (flash) size
- the maximum clock frequency
- the capability of the peripheral modules

5.2.1 CPU features

The CPU contains a small number of registers which can store temporary data words which are accessed directly by instructions, without the need of an address.
Some of these have a special role, like keeping the next instruction address or the return address of a subroutine.

The CPU contains an ALU (Arithmetic and Logic Unit), which in the simplest form performs addition, subtraction and boolean and shift operations.
In this case, more complex operations like multiplication and division are performed by software, by combining simpler operations.
More advanced CPUs can perform multiplication and division in a single instruction, and high-end CPUs have the hardware floating point capability.

5.2.2 RAM memory

Microcontrollers use static RAM (SRAM), which is fast and consumes little power, but is limited to smaller capacities than the dynamic RAM (DRAM) used in desktop computers.

RAM is volatile, which means that the data it contains are lost each time the power is switched off.

The fact of being volatile is a disvantage, but this memory is much faster than non-volatile memories in particular when writing.

In architectures where the data memory and the program memory are entirely separated ("Harward architectures"), the RAM is simply called "data memory".

5.2.3 FLASH memory

Microcontrollers need some non-volatile memory to have some program ready to execute immediately after power-up. Non volatile memory can be :

- pure ROM (Read Only Memory) also named OTP (One Time Programmable) memory
- EEPROM (Electrically Erasable and Programmable ROM) which can be written but is treated as ROM by the application programs
- FLASH memory, which is just a high density EEPROM where erasing is done one large blocks instead of single words

In modern microcontrollers, programs are generally stored in flash memory. The CPU can read this memory, but not write into it.
Writing requires either an external hardware or a dedicated peripheral, and takes much more time than a regular memory access.

A small amount of OTP-ROM is sometimes present to store the program which is used to write programs into the flash memory under control of the development computer. Such a program is called the bootloader.

5.2.4 Peripherals

The CPU communicates with the peripheral by means of specialized registers which have a dedicated address space besides the main memories.

There are many types of specialized peripherals, here are the most common.

GPIO: General Purpose Input-Output

Each pin involved in GPIO can be programmed as a logic input or a logic output, and then handles one bit at a time under program control.

Logic 1 or "true" is represented by a high voltage, close to the power supply voltage (most often 3.3 Volts), while logic 0 or "false" is represented by a voltage close to zero (ground).

GPIO pins can source or sink a limited current, 4 mA to 20 mA depending on the model

Serial communication principle

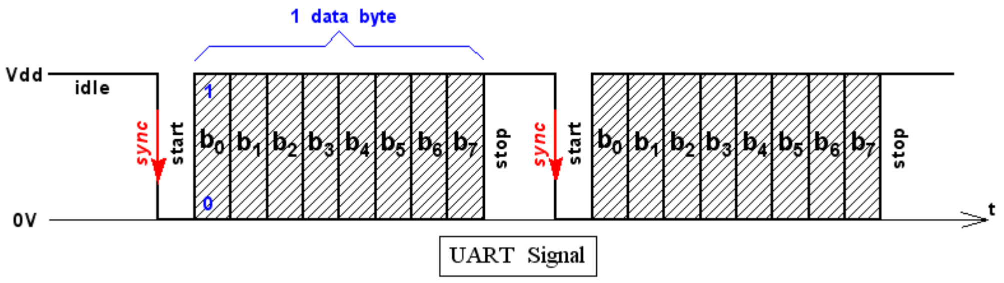
Serial communication involves the transmission of all the bits of each word in sequence, one after the other. The speed is reduced compared with parallel busses, but the cost of wiring is lower.

Some serial communication schemes transmit the MSB (most significant bit) first, some others transmit the LSB first.

		data wires	clock wires	select wires	total wires incl. gnd	typ. speed	protocol	range
UART	Universal Asynchronous Receiver-Transmitter	2	0	0	3	115200 Bauds	peer-to-peer	PCBs
I2C TWI	Inter-Integrated Circuit Two Wire Interface	1	1	0	3	400 kbits/s	master-slave	PCBs
SPI	Serial Peripheral Interface	2	1	1+	5	25 Mbits/s	master-slave	PCBs
USB	Universal Serial Bus	2	0	0	4	480 Mbits/s	master-slave	Desk
Ethernet	LAN (Local Area Network)	4	0	0	4	100 Mbits/s	peer-to-peer	Building

Serial asynchronous communication : UART

UART transmits blocks of one byte, framed by a "start" bit (always zero) and a "stop bit" (always 1).



The start bit leading edge causes the synchronisation of the receiver clock, which is used to sample the data bits. The synchronization is reset at the start of each byte

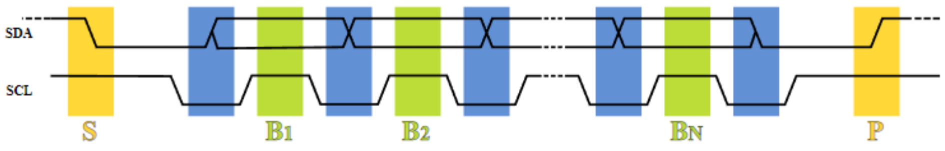
The receiver clock frequency must be configured to match the transmitter clock frequency with a sufficient accuracy (~2%).
This frequency is historically called the "Baud rate". The usual Baud rates values are such as 300 * 2^N.

No master is defined at the hardware level, this is left to the application software.

For increasing the range to building area, the UART signals are amplified to higher voltages (RS232) or differential signals (RS485). RS232 is obsolete but RS485 is still used for industrial equipments

Serial synchronous communication : I2C

I2C is sometimes named TWI for historical copyright reasons.



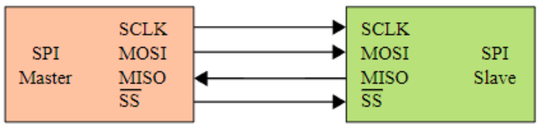
The master provides the clock signal on SCL, and the data line SDA is bidirectional.
The master creates special patterns S (Start) and P (stop) to mark the block limits.

The direction of transfer on the data line may be changed during a transaction, which involves a complex protocol.

The bus supports several slaves, thanks to an address field inserted by the master at the beginning of a transaction.

Serial synchronous communication : SPI

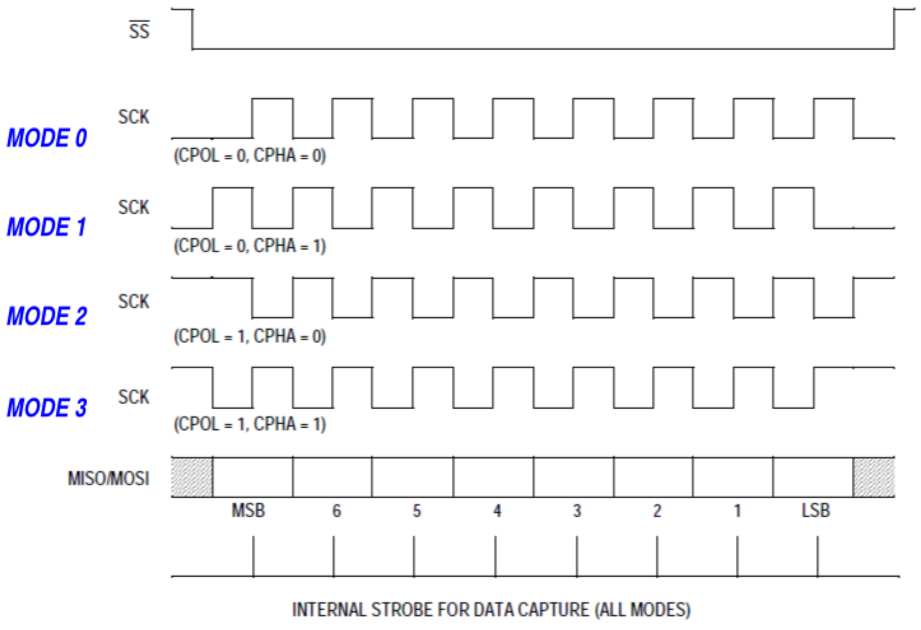
SPI is a very simple but loosely defined serial communication scheme



It uses 2 unidirectional data lines MOSI (Master Out Slave In) and MISO (Master In Slave Out).
There is one active-low SS (Slave Select) signal for each slave, which the master uses to select a slave before driving the clock line SCLK.

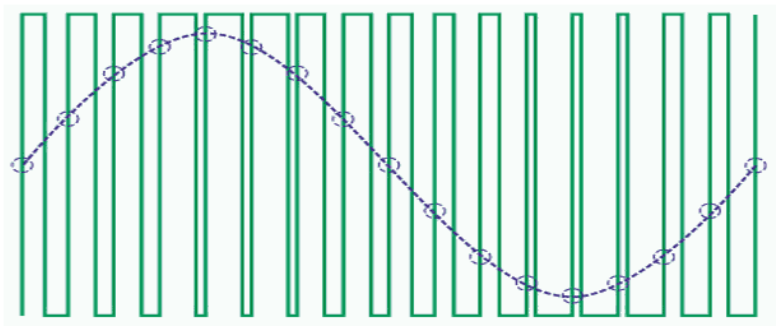
There are 4 modes described in the initial specification, but what really matters is the direction of the clock edge which is used to sample data at the receiving end of each data line.

- Modes 0 and 3: data sampled at the rising clock edge
- Modes 1 and 2: data sampled at the falling clock edge



TIMER: Time keeping and PWM signal generation

A timer is a digital binary counter driven by a periodic signal (the CPU clock or a signal derived from it by frequency division).
It provides a high resolution time measurement (microseconds) and can generate periodic binary signals of arbitrary period and arbitrary pulse width (period and pulse width being multiples of the clock period).



The pulse width may be modulated in real time to produce a PWM signal (Pulse Width Modulation).
The average value of the PWM signal is proportional to the function used to modulate the pulse width.
This makes a convenient way to generate a slowly varying analog signal using a single digital pin, and to control a power device with reduced losses compared to a continous control.

ADC and DAC

The ADC (Analog to Digital Converter) produces a binary number proportional to the voltage received on a pin.

The conversion is not continuous and must be triggered periodically by software, or directly by a timer.

This periodic conversion has the same limitations as any sampled signal processing, notably the input signal frequency should no exceed half the sampling frequency, otherwise the acquired signal is corrupted by aliasing (See chapter 4.4 The aliasing issue)

The DAC (Digital to Analog Converter) produces the inverse service. Notice that a timer in PWM mode also makes an economical DAC solution, but its output should be filtered to obtain a clean analog signal.

RTC : Real Time Clock

The RTC keeps the time and date even when the main power supply is off, having a separate power supply.
Like watches, the RTC has a very low power need, being able to run for years on a small battery.

The RTC modules frequently contain a small amount of memory called the backup RAM, which keep application data when the microcontroller is off.

5.2.5 Interrupts

The CPU may accept a hardware interrupt signal from some peripherals. When this happens, the CPU completes the current instruction and then leaves the current instruction sequence and jumps to a specific interrupt service routine subroutine or interrupt handler, in order to perform some action required by the peripheral which emitted the interrupt request.

When the end of the interrupt routine is reached, the CPU returns to the instruction sequence where it was interrupted. To ensure that the interrupted task is not disturbed by this process, a context saving sequence is strictly applied before entering the interrupt routine, and a matching context restoring sequence is applied upon return, by hardware and software combined.

Interrupt vectoring

An interrupt controller may be used to manage several sources of interrupt requests, and to associate a dedicated interrupt routine for each of these sources. This is called interrupt vectoring.

Interrupt priority

The interrupt controller manages simultaneous interrupt requests by following some priority rules. On the simplest microcontrollers, a fixed priority is assigned to each peripheral. On more advanced ones, the priority level of each source can be set by program.

Interrupt nesting

On the simplest microcontrollers, interrupts are disabled upon entry into an interrupt routine, but the code inside this routine may re-enable interrupts to permit the interruption of itself by another source, considered as deserving a quicker handling. In any case interrupts are automatically enabled after return from the interrupt routine.

On more advanced architecture, the interrupt controller may permit automatically the interruption of an interrupt routine by another interrupt of higher priority, this is called interrupt nesting.

Deadlock risk

If inside an interrupt routine, a function is called to perform some communication action which depends on a lower priority interrupt service, this action will never be completed and the system will be blocked. To avoid this, the developer should be aware of all possible sources of interrupt requests.

Multitasking

Interrupts are essentiel to the implementation of multitasking, in computers of any size as well as in microcontrollers. A periodic interrupt requested by a timer is theoretically sufficient to make a multitask system, but a more efficient handling of I/O streams is permitted by specific interrupts

- In personal computers, interrupts are managed by the operating system in a manner which is transparent for the application programs
- On microcontrollers, managing interrupts is the developer's business

5.3 Working with the Arduino

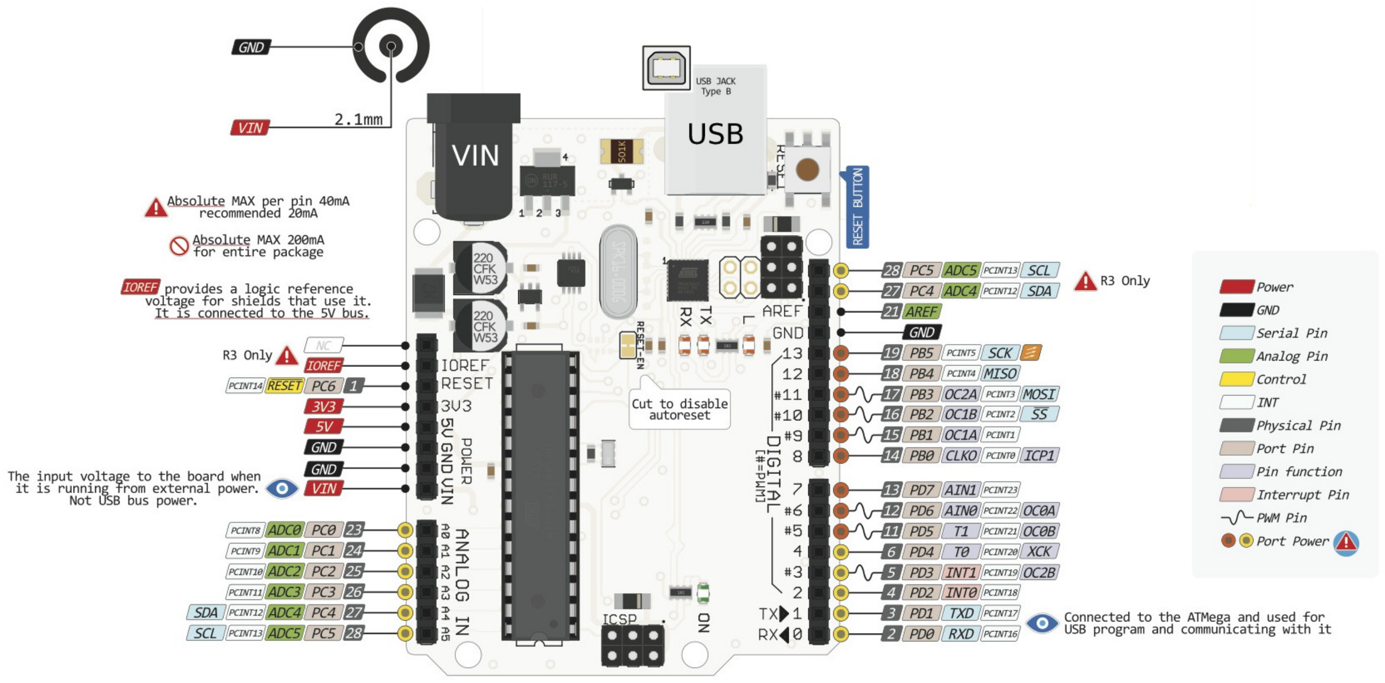
Arduino is the name of an Italian private company which made a successful attempt to enable non-professionals to learn electronics through embedded computing, and produce some home-made microcontroller based applications.

In this context, Arduino produced a minimal development platform, based on a moderate cost microcontroller board and a user-friendly free development software.
When people talk about "The Arduino" they refer to the board, or to the microcontroller itself.

While some choices made by Arduino are highly questionable, the benefit of this project is a worldwide acceleration of the DIY (Do-It-Yourself) microcontroller development activity. Notably many competing manufacturers put higher performance, lower cost boards on the market, while the open source community produced alternate software solutions.

5.3.1 The Arduino Uno hardware

The Uno is the first Arduino board and the most famous. While its technology was nearly outdated when it was introduced, it is still produced.



The board is equipped with two microcontrollers :

- the target microcontroller, which executes the user's programs and exposes 22 I/O pins for external wiring
- the USB gateway microcontroller, which acts like a bridge between the target microcontroller's UART and the host computer's USB port

On delivery, the target microcontroller flash memory contains already a small bootloader program, which is ready to receive program data via the UART and write them into the remaining space in the flash memory. The gateway microcontroller contains the USB-UART bridge software.

Why 2 microcontrollers? The USB communication protocol is extremely complex and having the USB port on the target microcontroller would not have left sufficient resources for the user programs. In the other hand, the UART-based bootloader is lightweight.

The target microcontroller is an ATmega328P made by ATMEL (now belonging to Microchip), with the following characteristics:

- 8-bit AVR RISC CPU
- 5 Volts power supply and I/O
- 16MHz clock
- 32 kbytes of flash memory
- 1 kbyte of EEPROM memory
- 2 kbytes of RAM memory
- Harvard architecture : Flash and RAM accessed in parallel
- Vectored interrupts (26 vectors) with fixed priority
- Three timers with 2 PWM channels each
- One SPI port, with an 8-bit shift register
- One UART
- One I2S "2-wire Serial Interface"
- One ADC, 10 bits resolution, 15 ksamp/s, 6 multiplexed inputs

The Uno board exposes a proprietary connector layout for which many accessory boards, called "shields", were developed.

The Uno board can be powered by the USB (5 Volts nominal, 4.5 Volts min.) or by an external supply from 6 to 14 Volts, thanks to a 5 Volts voltage regulator. The board also has a 3.3 Volts regulator for the needs of shields.

Note: Around year 2000 the standard supply voltage for microcontrollers was shifted from 5 Volts to 3.3 Volts, this change was also applied to every digital chip communicating with SPI, I2C, UART.
The ATmega328P could run from 3.3 Volts at a reduced clock frequency, but Arduino chose to stick to 5 Volts. For this reason, the accessory boards or "shields" frequently include voltage translator ICs to ensure compatibility with 3.3V chips.

5.3.2 The Arduino software

Arduino proposes an IDE (Integrated Development Environment) running on a host computer (Windows, Mac OS, Linux).
This IDE is free and open-source, being based on existing open-source projects, mainly the wiring IDE and the GCC compiler, which is a C/C++ toolchain.
The IDE contains the following software components :

- A simple text editor with syntax highlighting
- A build system which is aware of the Arduino libraries and performs some text pre-processing before calling the GCC compiler and linker
- An uploader utility, which communicates with the target microcontroller's bootloader via the USB link and the USB to UART bridge
- A serial monitor, which is a terminal allowing to exchange text messages with a user program running on the target microcontroller
- A serial plotter, which draws graphics from the data it receives from the target microcontroller

With the intention of keeping things apparently simple, the IDE ('legacy version') comes with the following limitations:

- Splitting the source code into more than one file is not encouraged
- Browsing the source code over multiple files is not supported, but a context sensitive help is available for Arduino library functions
- Build steps are hidden, unless switched to "verbose" in the preferences
- No incremental build
- No debugger

However, the system is more open that it seems at first sight. With some efforts it is possible to split the source code, use legacy C text formatting (snprintf), perform direct access to the peripheral registers, and use interrupts (all this thanks to the avr-libc library)

The C/C++ standards require a main() function with 2 optional arguments, returning an int.
This is not adapted to the microcontroller, where the program cannot receive any argument and must never return.
For this reason, the Arduino IDE expects the main program file to contain instead a setup() function and a loop() function.
The setup() function is called once, then the loop() function call is repeated indefinitely.
(These calls are included in a main() function created automatically by the Arduino build system).

Arduino jargon:

Arduino term	*actual meaning*
Arduino language	C++, with specific libraries
sketch	a project for Arduino, in fact a directory containing a single .ino source file and some documentation
sketchbook location	library search path root
file extension .ino	file extension .cpp
Verify	compile and link
Upload	compile and link then upload
analogRead()	start an ADC conversion, wait for completion, return value
analogWrite()	first time, start a timer as PWM generator on that pin, else update pulse width
attachInterrupt()	configure an external interrupt linked to one pin

5.3.3 The microcontroller as a PC peripheral

The USB to UART bridge which is on the Uno board is viewed by the host computer as a CDC (Communication Device Class) USB device.
This means that, beside the two possibilities offered by the Arduino IDE, any program running on the host computer may dialog with the target microcontroller, using the standard serial communication interface of the computer (the COM ports in Windows, the TTYs on Linux).

The Arduino IDE has two built-in tools for serial communication :

- the Serial Monitor, a text-only terminal
- the Serial Plotter, displaying curves according to numbers received from the microcontroller

To send a text command to the board, use the input area at the top of the serial monitor or at the bottom of the serial plotter. A single byte command may be sent, by choosing "No line ending" instead of "Newline".

The serial plotter has some interesting undocumented properties, here are some tips:

- to plot N curves :
    - send an initial text line with N words (each one beginning with a letter, not a digit), to be displayed as labels for the curves
    - then send text lines, each one containing N numbers (separated by spaces or commas),
      each line will append a new segment to each of the N curves, and increments X by one unit
- horizontal scale : the X unit is always 1/500th of the window width, the first 500 text lines fill the window, then the curves are scrolled to the left
- vertical scale : is automatically adjusted,
  to keep it stable, plot two additional curves with constant Ymin and Ymax values, then they will "frame" the graphics
- to plot variables in real time, send lines periodically with a constant period, (delay function, or better, timer interrupt)
- to plot calculated curves, or previously acquired measurements, send exactly 500 text lines, then pause to freeze the graphics

Notes:

- The serial monitor and the serial plotter cannot be active simultaeously
- Each time the serial monitor or the serial plotter is started, the target microcontroller is reset.

5.3.4 Low level programming

"Low level" means "close to the hardware", in a layered model of application development.
The complications of this level are "abstracted" (hidden) by an intermediate layer of apparently simple functions ("drivers"), but it is sometimes necessary to be aware of the hidden details. The first step is to get familiar with the target microcontroller reference manual

GPIO direct access

Pin direction may be configured using the pinMode() function, writing and reading can be done one pin at a time with digitalWrite() and digitalRead()

By talking directly to the GPIO peripheral, it is possible to manipulate several pins simultaneously (up to 8).

For this it is worth noting that the I/O pins are grouped in GPIO ports, each one containing 8 pins.
In the case of the Uno board, three ports PB, PC and PD are used, the correspondance between the GPIO port numbering and the Arduino pin numbering is documented on the Arduino UNO schematic and the pinout diagram above.

Each port exposes three 8-bit registers, DDRx (the direction register), PORTx (the output register), PINx (the input register), x being B, C or D

In order to facilitate the access to these registers from a C/C++ program, symbols for every peripheral register are defined in file iom328p.h which is included by io.h which is included by Arduino.h which is included automatically in every Arduino "sketch".

Timer interrupt and time measurement

Function millis() provides a value of the duration elapsed since the last CPU reset, in ms. (It overflows approximately every 50 days).

On the Uno, Timer 0 is configured for providing this service. With a prescaled clock of 16000000/64 = 250000 Hz, its internal counter measures time with a 4 us resolution, and with its 8-bit capacity it overflows every 1.024 ms.
In order to keep time on a longer scale, an overflow interrupt is configured to count the overflow events, and compute time in milliseconds from the overflows count and the timer internal counter.

Function delay() just sets an arrival date (millis() + duration) and loops waiting for reaching this date.
In multitasking applications, such a waiting loop is merged in the main loop in order to perform other tasks while waiting.

In the other hand, function delayMicroseconds() is based on an estimate of the execution duration of a repeated instructions sequence, it is worth using only for short durations (less than 1 ms).

Find file writing.c in the Arduino installation directory for all details

PWM outputs

The analogWrite() function starts a PWM source on one of the 6 PWM-capable pins, which is bound to one of the three timers of the Uno.
This function allows to vary the duty cycle from 0 to 100% according to an 8-bit argument (0 to 255), but does not allow any control over the frequency.

	digital pin	period
timer 0	5	1024 us
timer 0	6	1024 us
timer 1	9	2040 us
timer 1	10	2040 us
timer 2	3	2040 us
timer 2	11	2040 us

Note: timer 0 is used for two purposes, which are compatible.

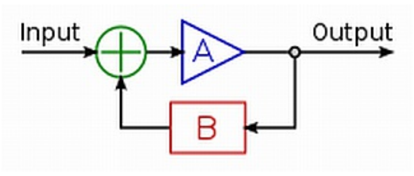
More details in document Secrets of Arduino PWM

Document links :

- en.wikipedia.org/wiki/Microcontroller
- C language for beginners
- C++ language for beginners
- en.wikipedia.org/wiki/Pulse-width_modulation
- docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm
- en.wikipedia.org/wiki/Serial_Peripheral_Interface
- en.wikipedia.org/wiki/I²C
- en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
- en.wikipedia.org/wiki/UTF-8
- en.wikipedia.org/wiki/Binary_number
- en.wikipedia.org/wiki/Hexadecimal
- en.wikipedia.org/wiki/Two%27s_complement
- Arduino UNO schematic
- ATmega328
- www.nongnu.org/avr-libc/user-manual/modules.html

6. The Feedback

Feedback systems are also known as closed-loop systems, where some output signal is fed back to a control input.

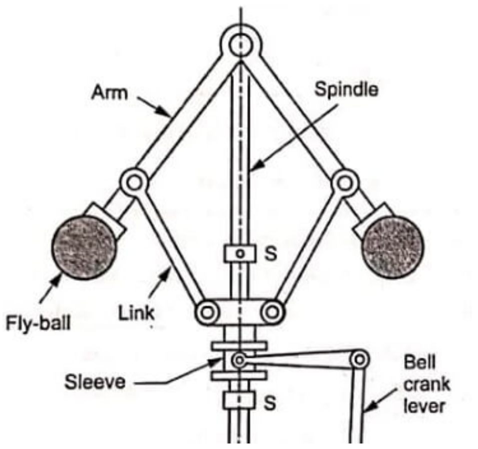


Feedback is essential to control theory and control engineering, nevertheless it also exists in nature.

This chapter presents feedback examples based on analog circuits, but mixed analog-digital feedback systems including control by software are current.

6.1 Negative feedback

The goal of a negative feedback loop (also known as servo, regulator or governor) is to keep the ouput of the system in accordance with an external input signal (set value), even in the presence of random perturbations

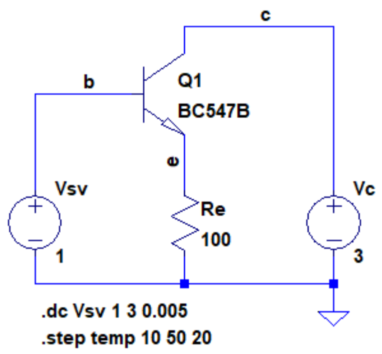


In a negative feedback loop, the feedback signal is subtracted from the from the set value, then the difference (error signal) is amplified to control the output

A greater amplification gain will provide a better rejection of the perturbations and a better accuracy of control, but an excess of gain can make the system unstable.

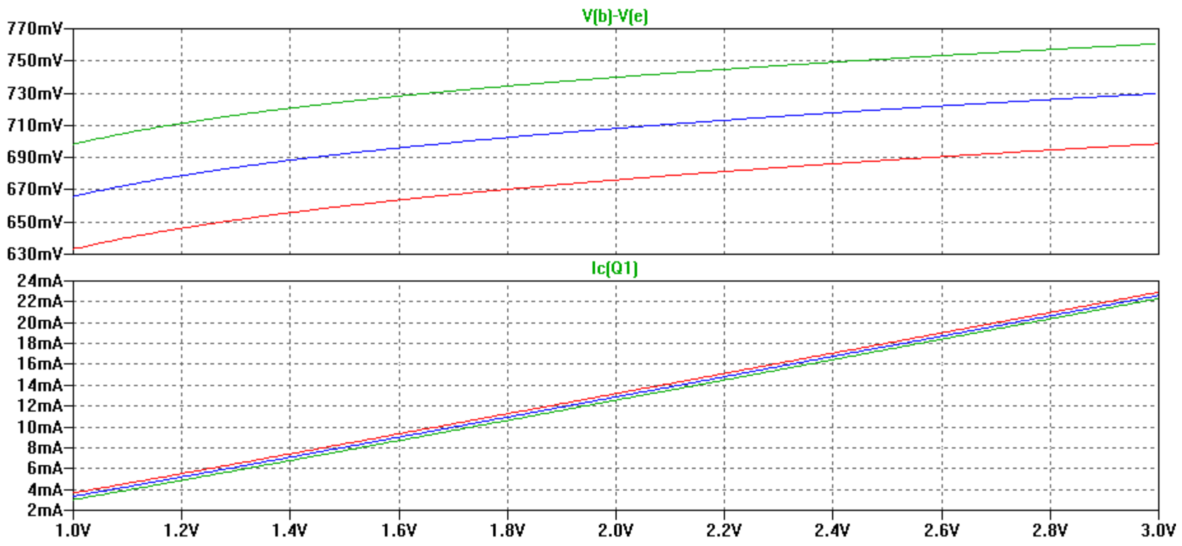
6.1.1 Regulated current source

The collector current Ic of a bipolar transistor is controlled by the base-to-emitter voltage Vbe with a very high gain.
But this control is non-linear and the current also strongly depends on the temperature and the process variations, in such a way that controlling the collector current in open loop is not feasible.



With a high gain transistor, the emitter current is nearly equal to the collector current, so the voltage across Re can be used as a feedback signal. In this circuit, Vbe is the difference between Vsv (the set value) and the voltage across Re, so Vbe is the error signal which will be amplified by the transistor.

The input source Vsv is swept from 1 Volt to 3 Volts, and this is repeated for temperatures 10°C, 30°C and 50°C.



Results: the output current is controlled linearly by the set value, with a constant slope close to 10mA/V, consistent with the value of Re, the temperature variations cause a large offset of Vbe but a moderate shift of Ic.

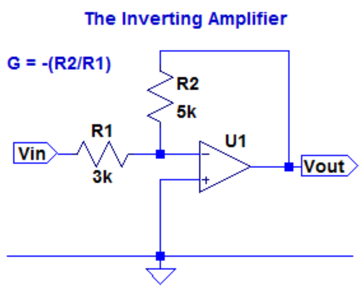
LTSPICE source file

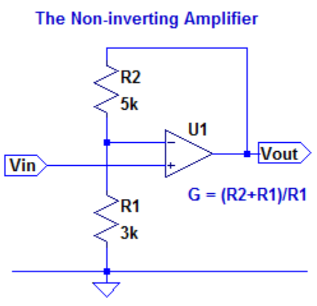
6.1.2 Operational amplifier basic circuits

The operational amplifier is a differential amplifier, built as an integrated circuit containing many transistors. It amplifies the potential difference between its inputs with a very high gain, in such a way that when the feedback is working, this difference is so small that it can be neglected.

The name "operational" comes from the fact that this amplifier was created for making the "operator modules" in the analog calculators.

The practical behaviour of the operational amplifier is entirely determined by the feedback circuit, which allows a great number of applications, the best known being the inverting amplifier and the non-inverting amplifier.





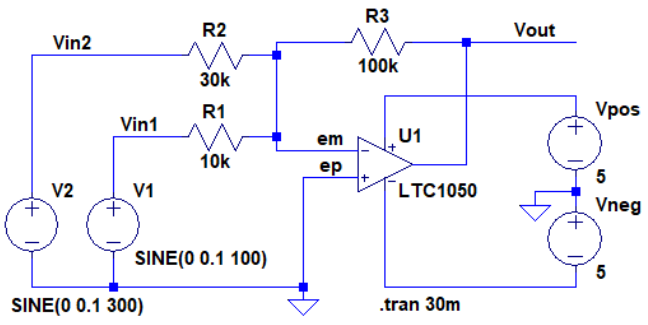
Note that if R2=0 in the non-inverting circuit, the gain G is 1 and R1 can be removed, this is the "voltage follower" circuit.

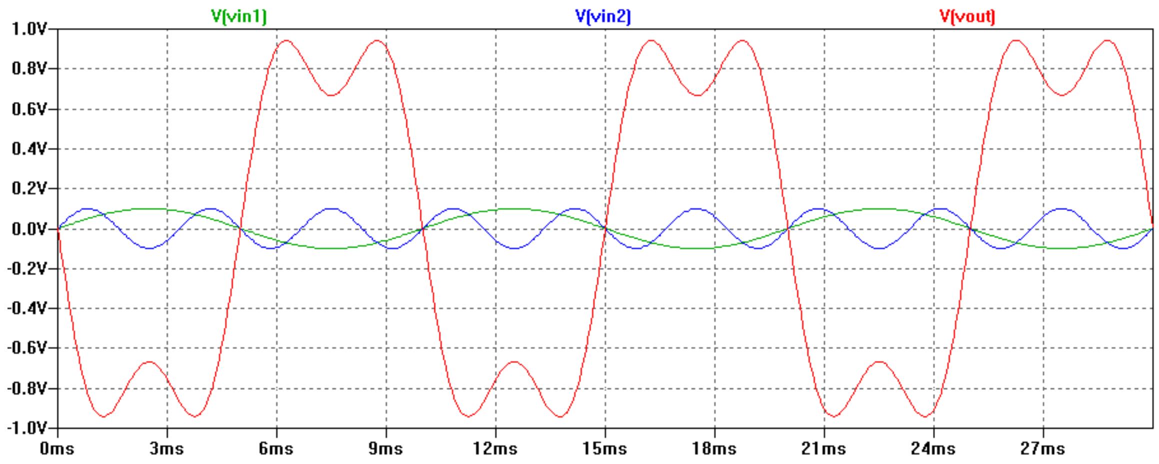
(Drawings from video opamp analogy live demo)

The next example is a weighted adding-inverting circuit, it produces the sum of two signals with different negative gains (-10.0 and -3.33 in the example).

To predict the output voltage, one has to consider that :

- the input current of the opamp is negligible, so the sum of the currents in R1, R2, R3 is zero
- the differential input voltage is negligible, so the voltage at em is zero





LTSPICE source file

6.2 Positive feedback

Ideally, a sufficient positive feedback creates an unbounded increase of the signal,
in practice the amplifying element will reach its limit, which is called saturation or clipping

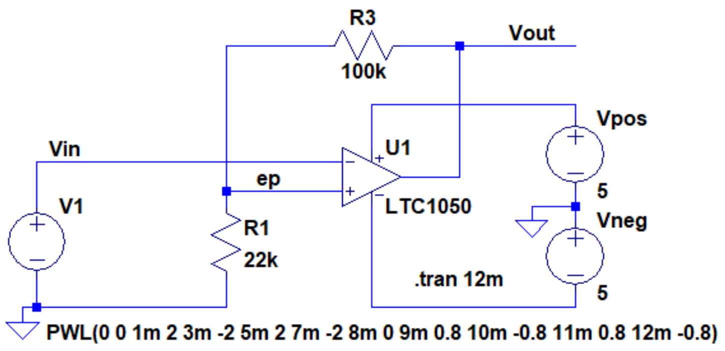
- if the feedback path includes the DC component, the amplifier will get saturated either at its maximum output or at its minimum output, until an exterior event causes it to flip the other way
- if the feedback path has a narrow bandwidth, the amplifier may become oscillating with an increased amplitude, which will be limited by clipping

6.2.1 Schmitt trigger

A trigger is a circuit which has only 2 stable output states, it is useful for processing input signals for digital circuits.

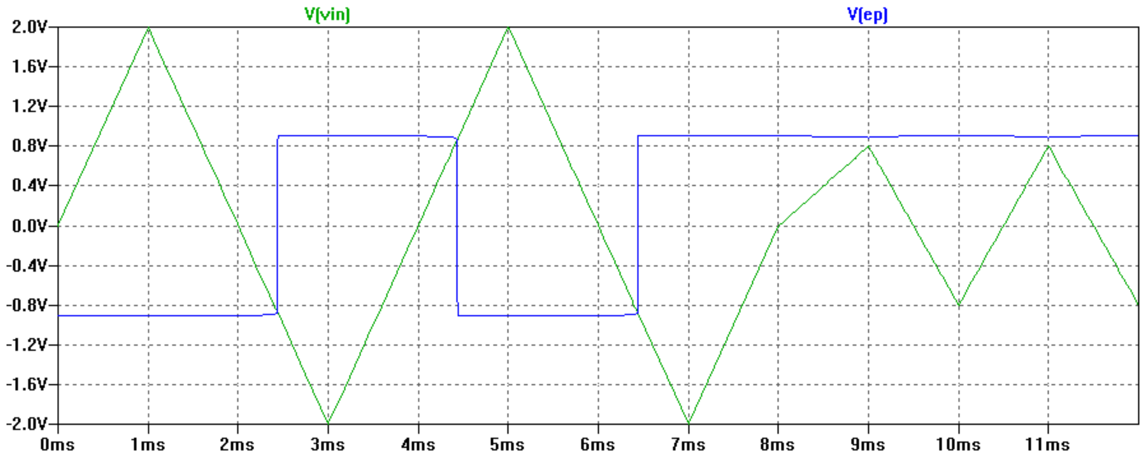
The Schmitt trigger is an amplifier with an attenuated DC feedback, in such a way that its flipping threshold (which is voltage at node ep) changes according to its output state.

Such a behaviour is called hysteresis.



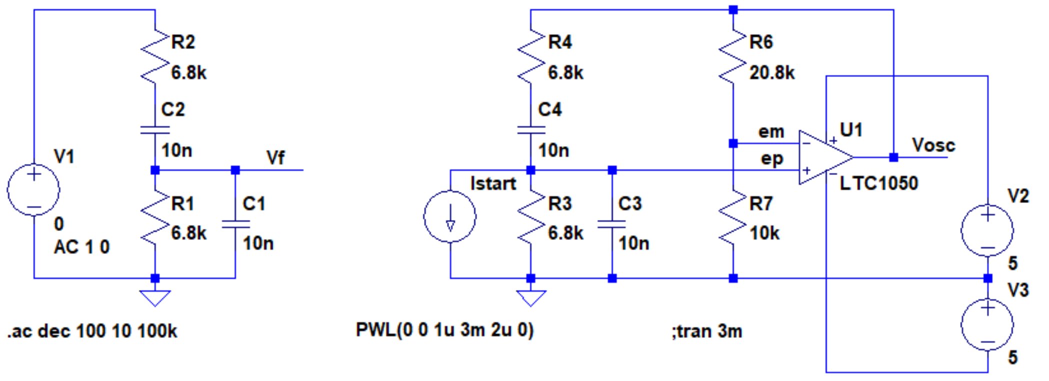
R3
100k
Vout
Vpos
Vin
U1
ep
LTC1050
5
Vneg
V1
R1
22k
.tran 12m
5
PWL(0 0 1m 2 3m -2 5m 2 7m -2 8m 0 9m 0.8 10m -0.8 11m 0.8 12m -0.8)

In the example, the thresholds are approximately -0.9 V and 0.9 V. It is observed that when the input signal does not cross these thresholds, the trigger ignores it.

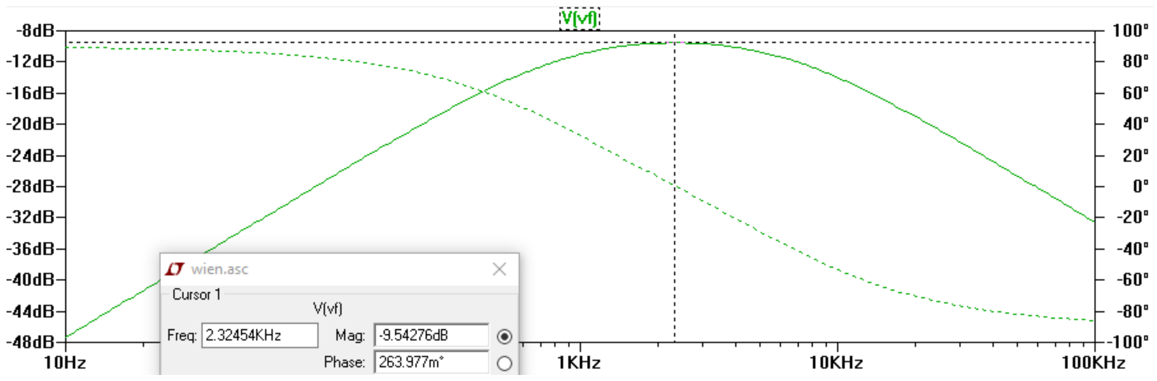


LTSPICE source file
LTSPICE source, same with single polarity power supply

6.2.2 Wien bridge Oscillator



The Wien bandpass filter (left drawing) will be used in the feedback loop, with the goal of making an oscillator producing a sinewave. A preliminary AC simulation will be used to check the behaviour of the filter alone, made of 2 identical resistors R1 and R2 and 2 identical capacitors C1 and C2.

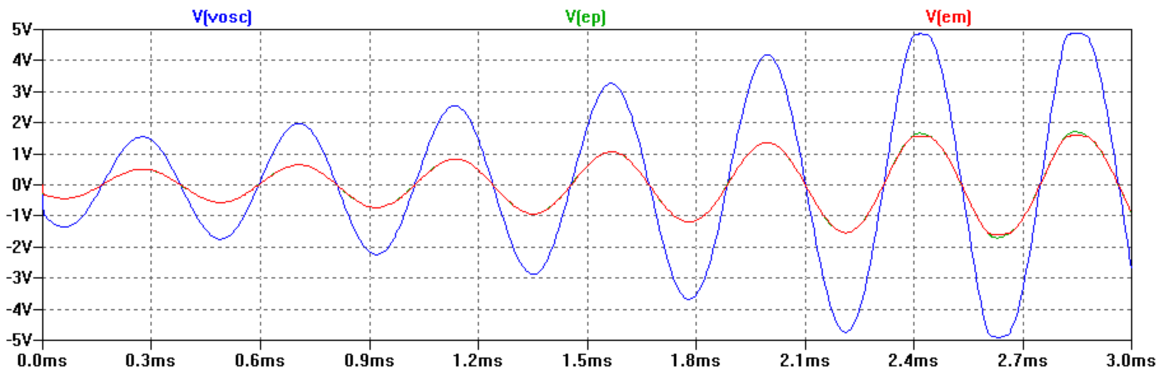


Freq: 2.32454KHz Mag: -9.54276dB
Phase: 263.977m°

The AC simulation shows that the filter has a maximum output of -9.54dB at 2.324 kHz, with a phase close to 0°. The theoretical output is 1/3, which is -9.5424 dB in good agreement with the simulation.

In order to start the oscillation, the amplifier must have a gain greater than 3, but not too much to avoid saturation.
In the example, this is done by combining an operational amplifier with a negative feedback provided by R6 and R7.

The simulator tends to rest on a metastable state instead of oscillating (this does not happen in real life).
To start the oscillation, a current source Istart provides a short pulse.



An oscillation is observed at the predicted frequency, with an amplitude increasing until saturation.
The saturation causes a distortion of the signal, which could be reduced by means of a amplitude regulation loop.

LTSPICE source file

Theory links :

- en.wikipedia.org/wiki/Control_theory
- en.wikipedia.org/wiki/Operational_amplifier
- https://youtu.be/GMBqciO8fKY opamp analogy live demo
- en.wikipedia.org/wiki/Schmitt_trigger
- en.wikipedia.org/wiki/Wien_bridge_oscillator
- https://youtu.be/-LlhZCc0B6U an example of analog+software loop + arduino code
- TLC2201 operational amplifier datasheet

Table of Contents