calc.pw

Password Calculation

with Arduino

Kenneth “Kenny” Newwood

E-Mail: kenneth@newwood.de

Jabber: cware@cwar.es

Twitter: @weizenspreu

Website: http://weizenspr.eu

Who's the speaker?

Agenda

What's the problem?

What's the idea?

What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

Agenda

What's the problem?

What's the idea?

What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

What's the problem?

one password per service is the best choice,

but: **remembering passwords is difficult**

password schemes simplify password memorization – **sometimes**

password databases simplify password memorization – but they can **get lost or stolen**

What's the problem?

exemplary *password schemes:*

Google password scheme[1]:

- select simple sentence
- remove spaces
- replace characters with numbers and special chars

[1] http://www.google.com/goodtoknow/online-safety/passwords/

What's the problem?

exemplary *password schemes:*

Prefixed password scheme[2]:

- select secure password prefix
- choose service-dependent password suffix (e.g. “ebay”)
- append suffix to prefix

[2] http://passwordadvisor.com/TipsUsers.aspx

What's the problem?

exemplary *password schemes:*

XKCD password scheme[3]:

- select some random English words
- append words to each other

[3] http://xkcd.com/936/

What's the problem?

password databases:

master password to encrypt other passwords

have to be available at all times[4]

can get **lost** (HDD crash) or **stolen** (trojan horse)

[4] http://bitbucket.org/HexRx/kpdatasave

Agenda

What's the problem?

What's the idea?

What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

What's the idea?

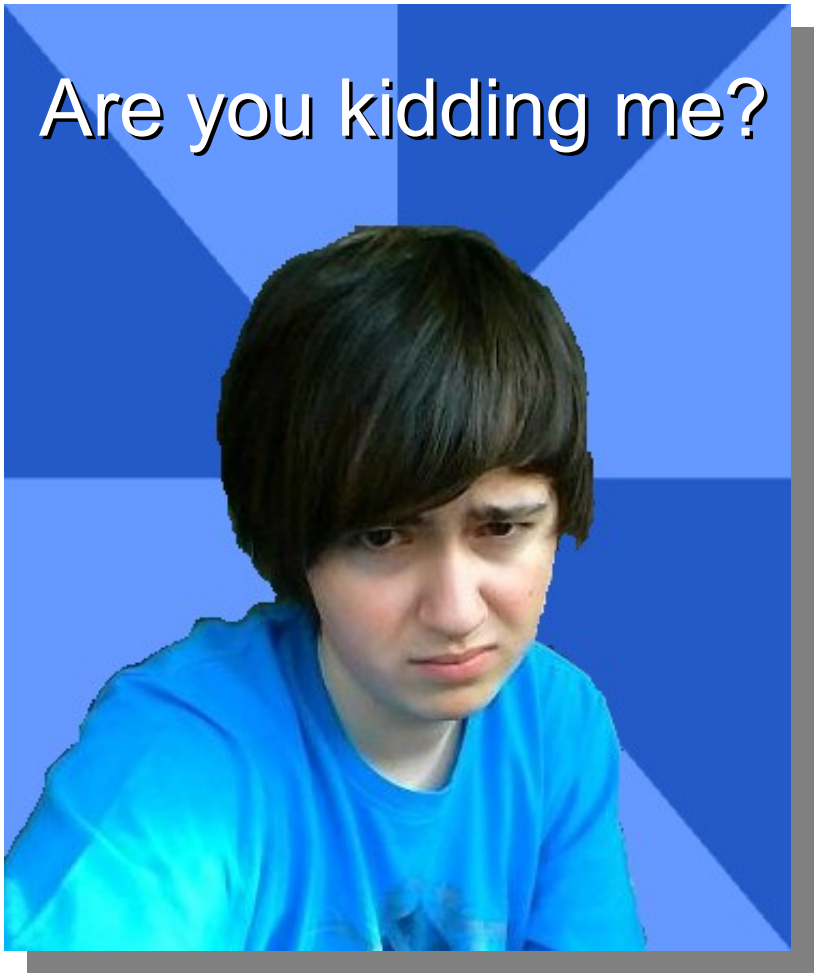
simplify password handling

solve password memorization problem

prevent password loss and theft

make it open source so everyone can use it

What's the idea?



What's the idea?

calculate passwords cryptographically

use secure master password for **strength**

use service information for **memorability**

use public algorithms for **reproducibility**

use dedicated hardware for **security**

Agenda

What's the problem?

What's the idea?

What's the solution?

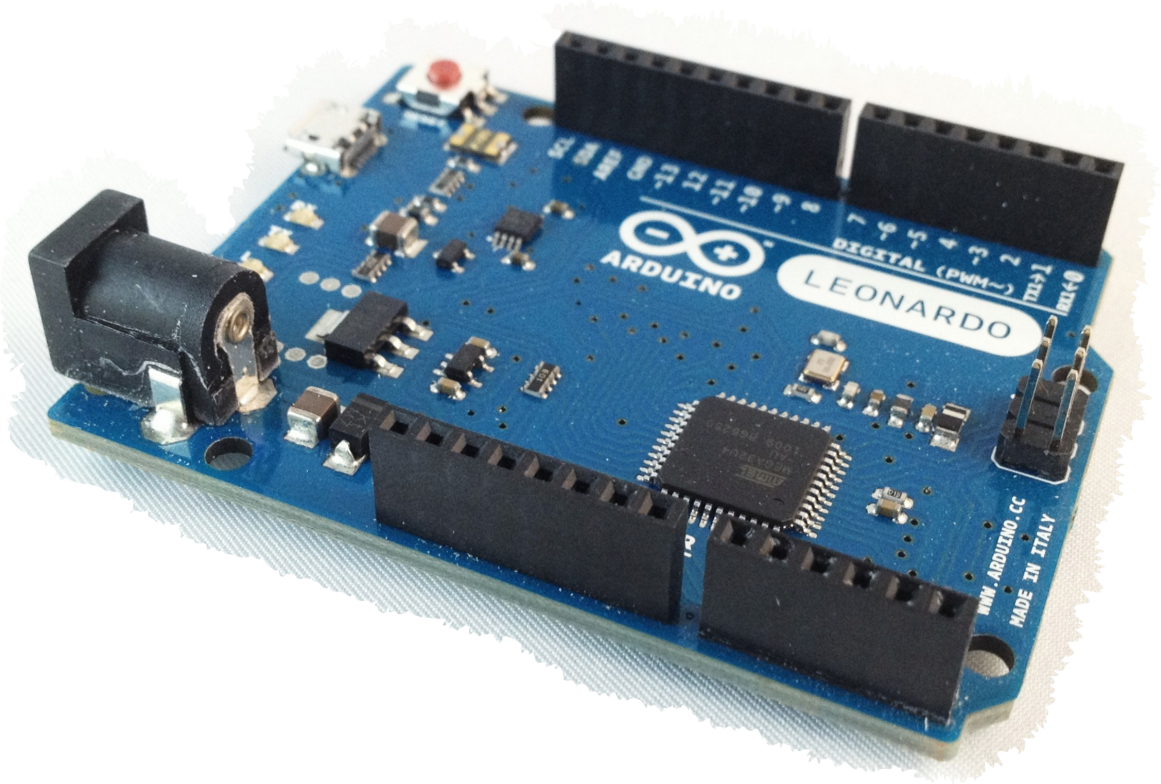
How does it work?

What're the pitfalls?

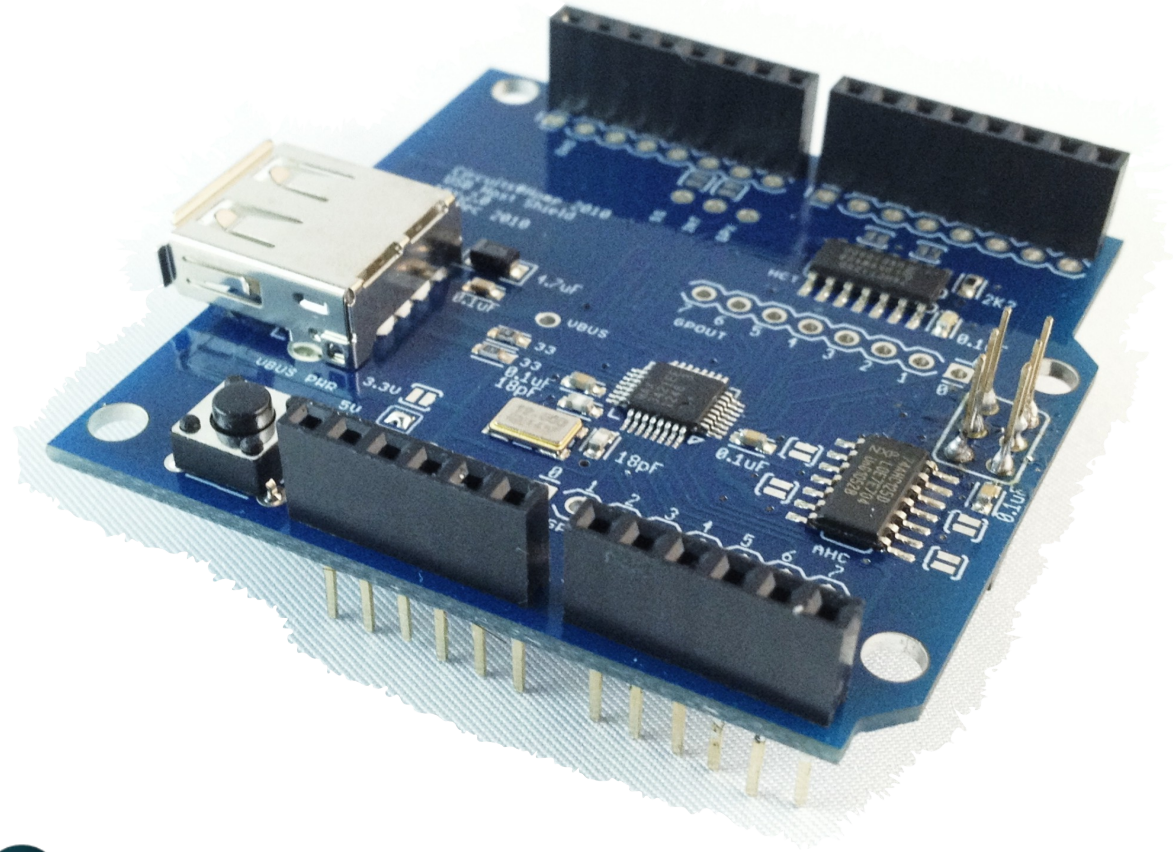
Where do I find more?

What's the solution?

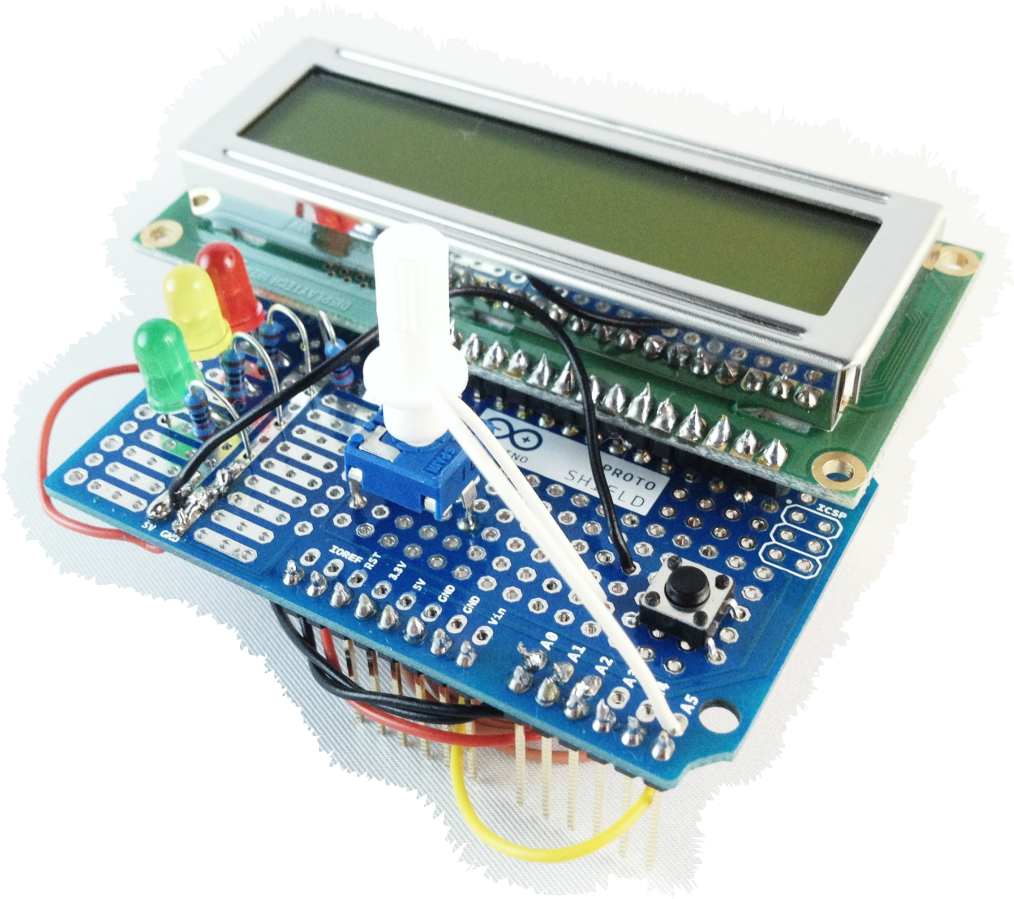
What's the solution?



What's the solution?



What's the solution?



What's the solution?

Arduino Leonardo as hardware basis

(processor, memory, keyboard emulation, etc.)

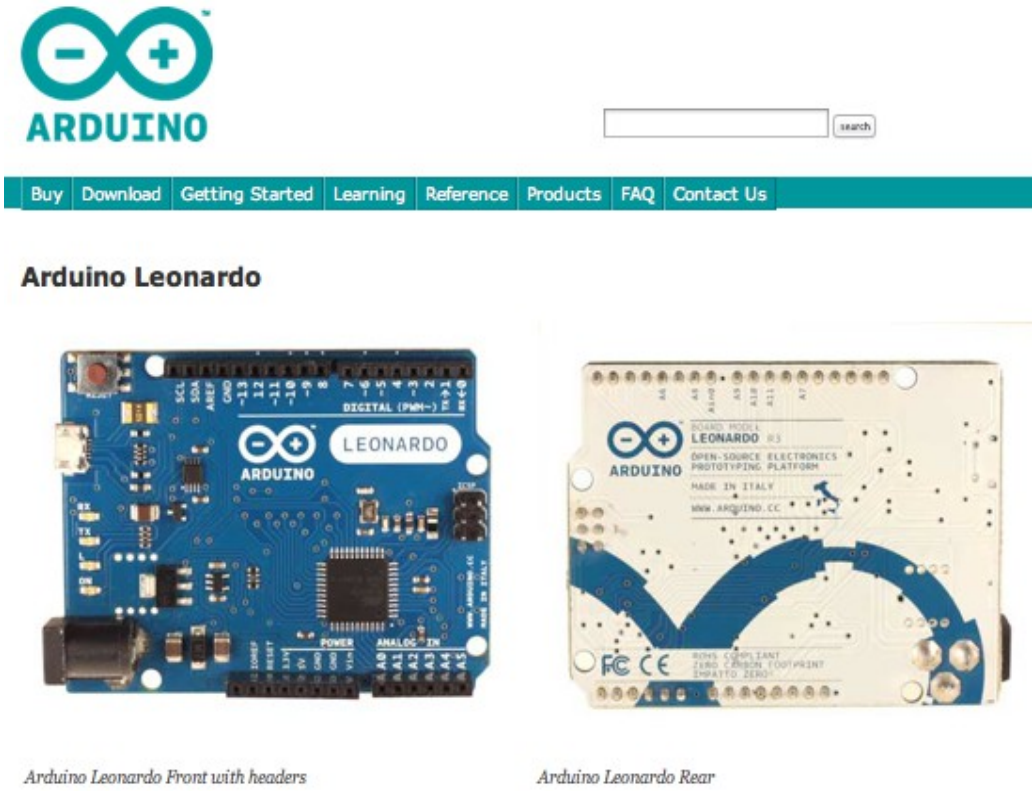
USB Host Shield to read keyboard input

(shield by *Circuits@Home* is best supported)

additional stuff as output

(*Arduino Proto Shield* is great)

What's the solution?



(www.arduino.cc)

What's the solution?

(www.lipoly.de)

What's the solution?

Arduino Proto Shield Rev3 (assembled)

Art.Nr.:	EXP-R08-020
Lagerbestand:	26 Stück

14,28 EUR

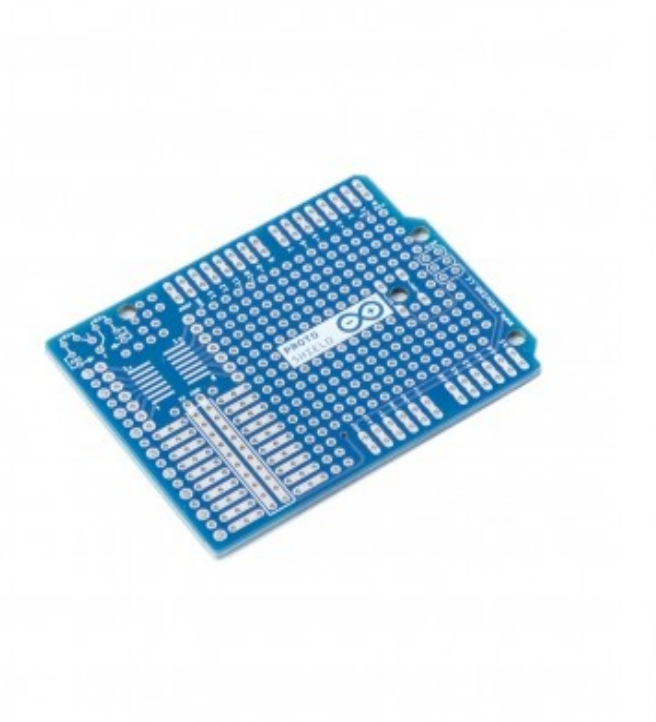
inkl. 19% MwSt. zzgl. Versand

1 In den Warenkorb

Auf den Merkzettel

(www.exp-tech.de)

What's the solution?



Arduino Proto Shield PCB Rev3

Art.Nr.:	EXP-R08-027
Lagerbestand:	109 Stück

3,57 EUR

inkl. 19% MwSt. zzgl. Versand

1 In den Warenkorb

Auf den Merkzettel

(www.exp-tech.de)

Agenda

What's the problem?

What's the idea?

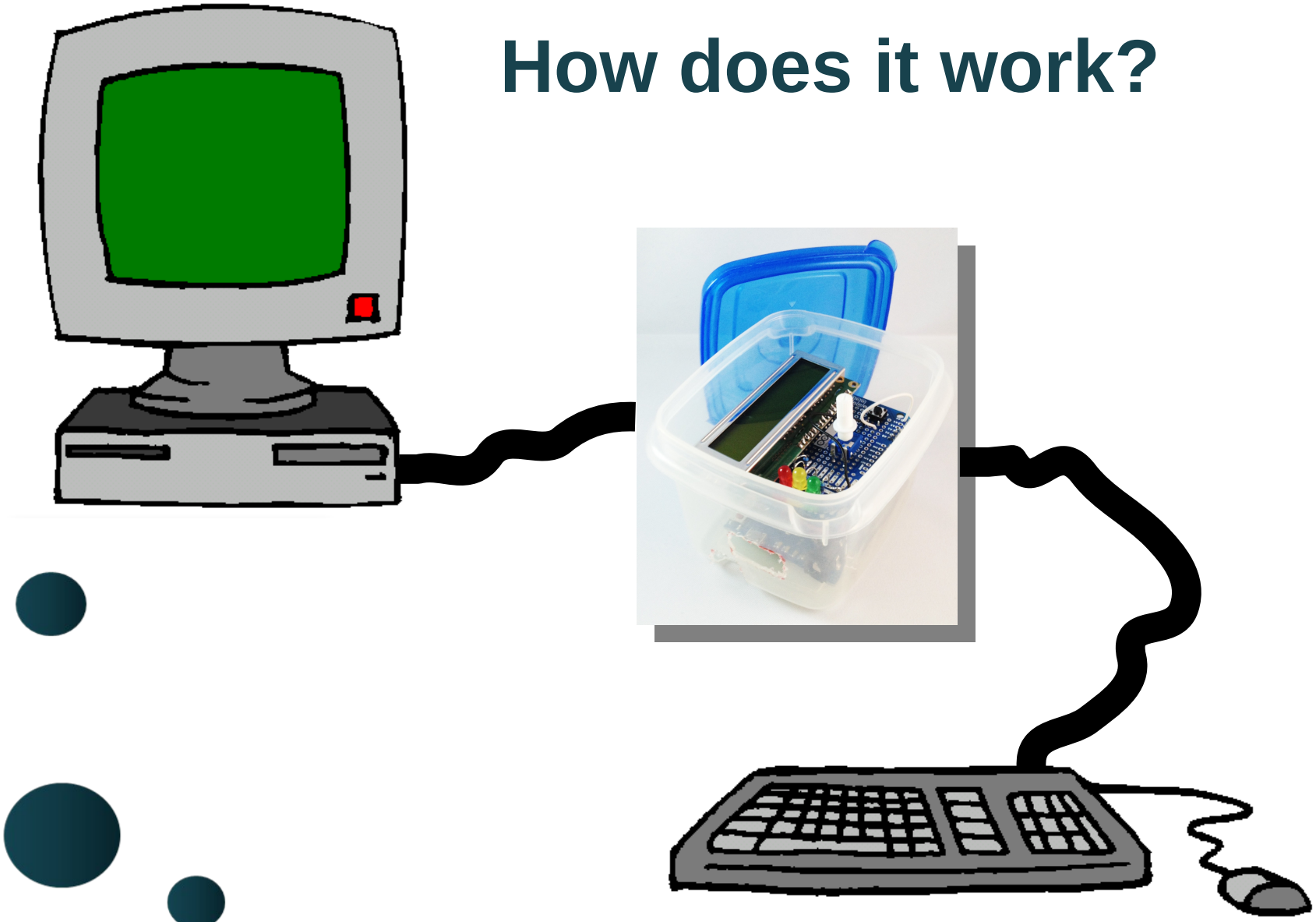
What's the solution?

How does it work?

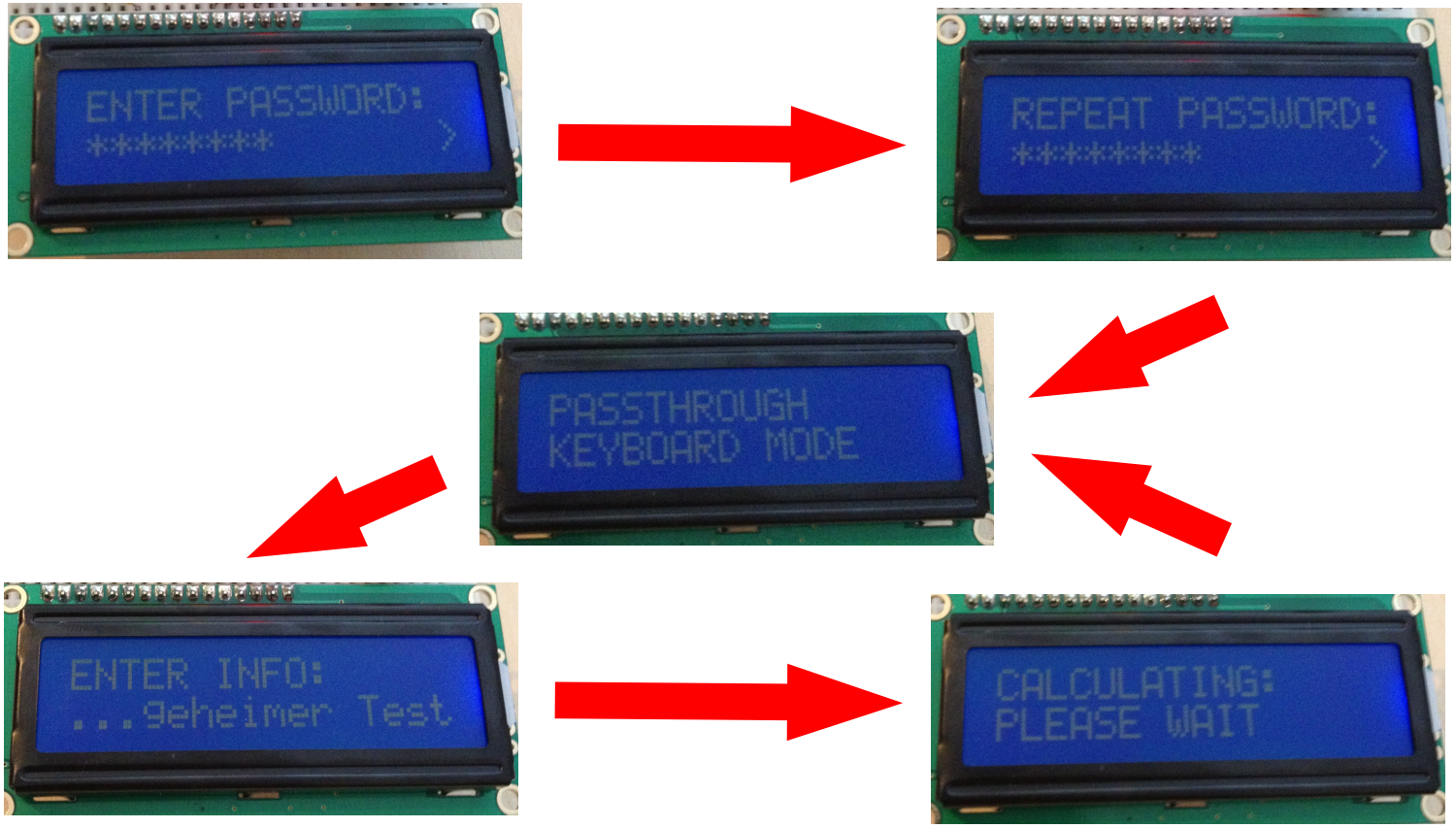
What're the pitfalls?

Where do I find more?

How does it work?



How does it work?



How does it work?

connects **between** keyboard and computer

enter master password on boot-up (twice)

switch to passthrough keyboard mode

activate password mode (Ctrl+Esc)

enter service information (Enter)

calculate password

switch back to passthrough keyboard mode

How does it work?

hash() = **SHA-1**

hmac() = HMAC-**SHA-1**

crypt() = **RC4**-drop1024

Magic(Information, Masterpassword)* =

hmacPass = hmac(Information, Masterpassword)

hmacInfo[i=0] = hmac(hash(hmacPass), Information)

hmacInfo[i=1..2] = hmac(hmacInfo[i-1], Information)

Password = base64(crypt(hmacInfo, hmacPass))

(* simplified)

How does it work?

?

define length of generated password (max. 50)

!

define set of possible specials characters

#

activate check for alpha-numerics

How does it work?

SomeINFO

SomeINFO?25

SomeINFO!+-*/

#SomeINFO

SomeINFO?25!+-*/

#SomeINFO?25!+-*/

Agenda

What's the problem?

What's the idea?

What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

What're the pitfalls?

keyboards are nasty little beasts

data flash memory limitations (1kb)

random access memory limitations (2.5kb)

processing speed limitations (16MHz)

program flash memory limitations (28kb)

What're the pitfalls?

lots of different keyboard layouts

(QWERTZ DE, QWERTZ CH, QWERTZ DK, QWERTY UK, QWERTY US, AZERTY FR, AZERTY BE, Mac/Windows, etc.)

not enough **program flash** to store all layouts

not enough **data flash** to store all layouts

solution: store one layout and reflash if needed

What're the pitfalls?

```
HEADER DONE
CHECKSUM DONE = 6553740
LENGTH DONE = 130
CALCULATED CHECKSUM = 6553740
FLASH DONE
```

qwertz_de.txt

```
7700EB 00D781 040061 040241 050062 050242
070064 070244 080065 080245 090066 090246
0B0068 0B0248 0C0069 0C0249 0D006A 0D024A
0F006C 0F024C 10006D 1001B5 10024D 11006E
12024F 130070 130250 140071 140140 140251
160073 160253 170074 170254 180075 180255
190076 190256 1A0077 1A0257 1B0078 1B0258 1C007A 1C025A
1D0079 1D0259 1E0031 1E0221 1F0032 1F01B2 1F0222 200033
2001B3 2002A7 210034 210224 220035 220225 230036 230226
240037 24017B 24022F 250038 25015B 250228 260039 26015D
260229 270030 27017D 27023D 28800A 28800D 2A8008 2C8020
2D00DF 2D015C 2D023F 2E00B4 2E0260 2F00FC 2F02DC 30002B
30017E 30022A 310023 310227 320023 320227 3300F6 3302D6
3400E4 3402C4 35005E 3502B0 36002C 36023B 37002E 37023A
38002D 38025F 54402F 55402A 56402D 57402B 58800A 594031
5A4032 5B4033 5C4034 5D4035 5E4036 5F4037 604038 614039
624030 63402E 64003C 64017C 64023E
```

What're the pitfalls?

limited **RAM** complicates memory handling

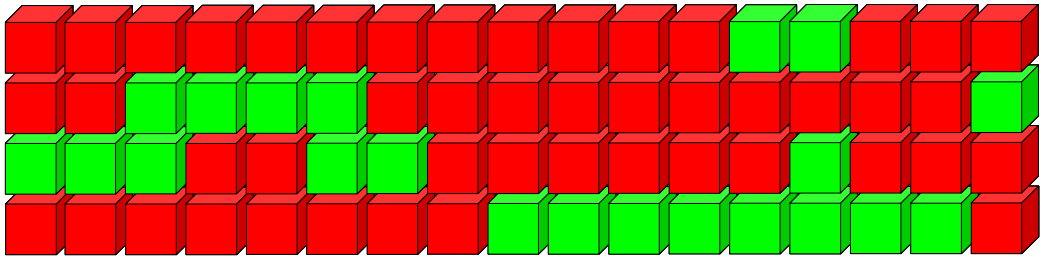
dynamic memory allocation is a bad idea

leads to fragmentation & potentially to corruption

solution: wrote own memory manager

- define size of handled memory
- define max number of possible memory chunks
- relocate memory whenever a chunk is freed

What're the pitfalls?



What're the pitfalls?

limited **program flash** is biggest problem

library of **USB Host Shield** grows steadily

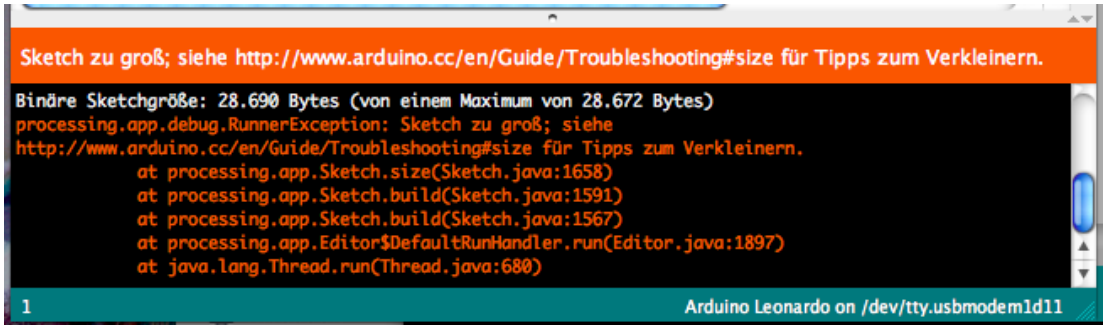
several options at the moment:

- replace USB Host Shield with other technology
  *(integrate keyboard, integrate **into** keyboard)*
- move calc.pw code out of program flash => HARVARD

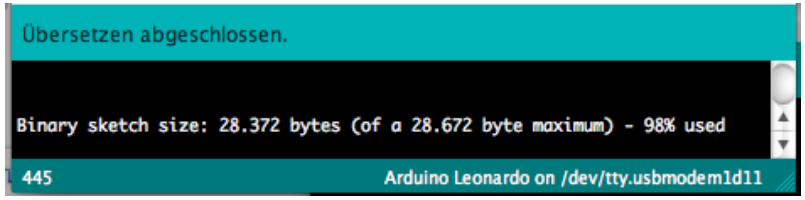
  (interpreter + code in external EEPROM = ArduROAM)

What're the pitfalls?

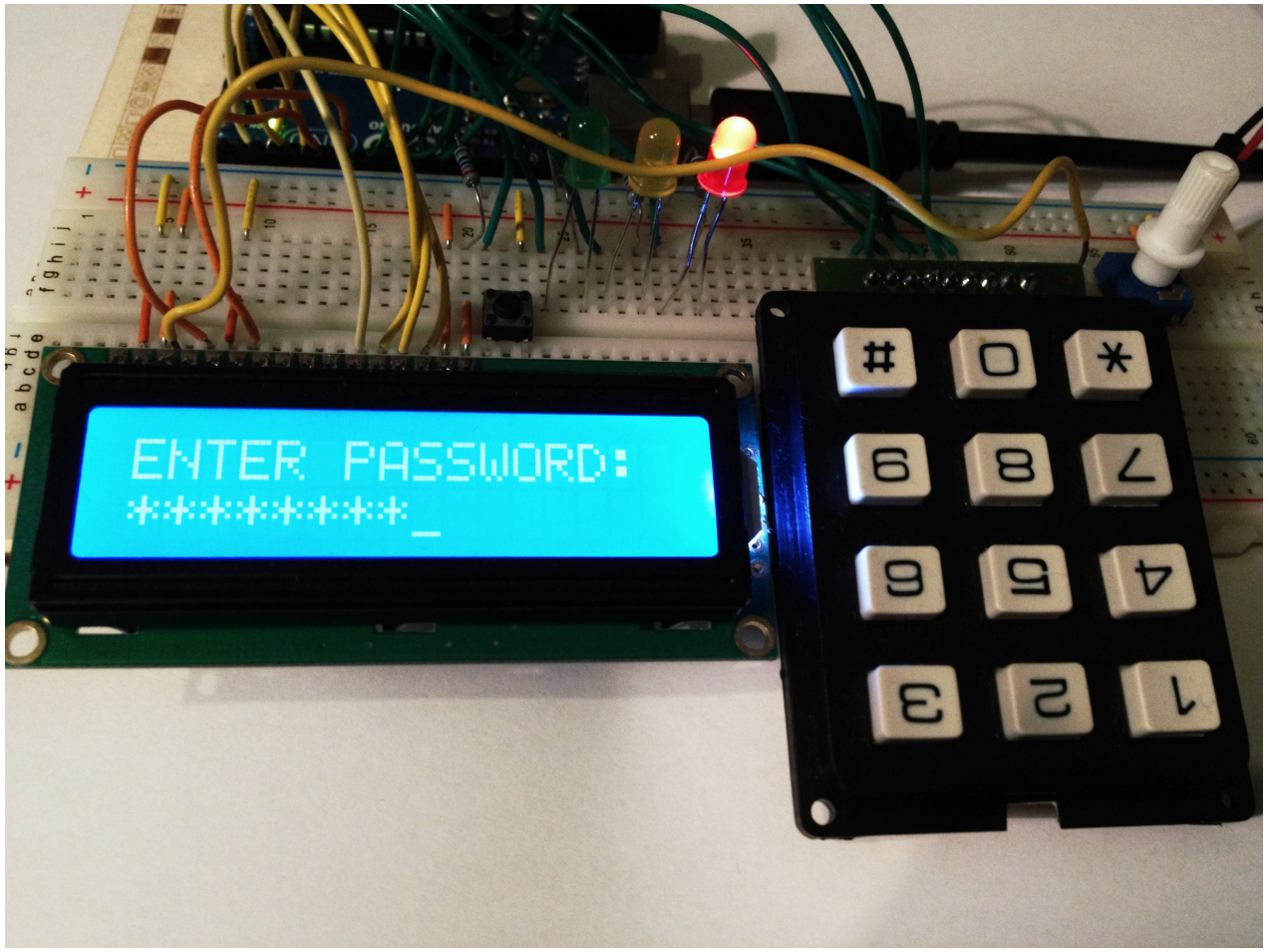
Arduino 1.0.5



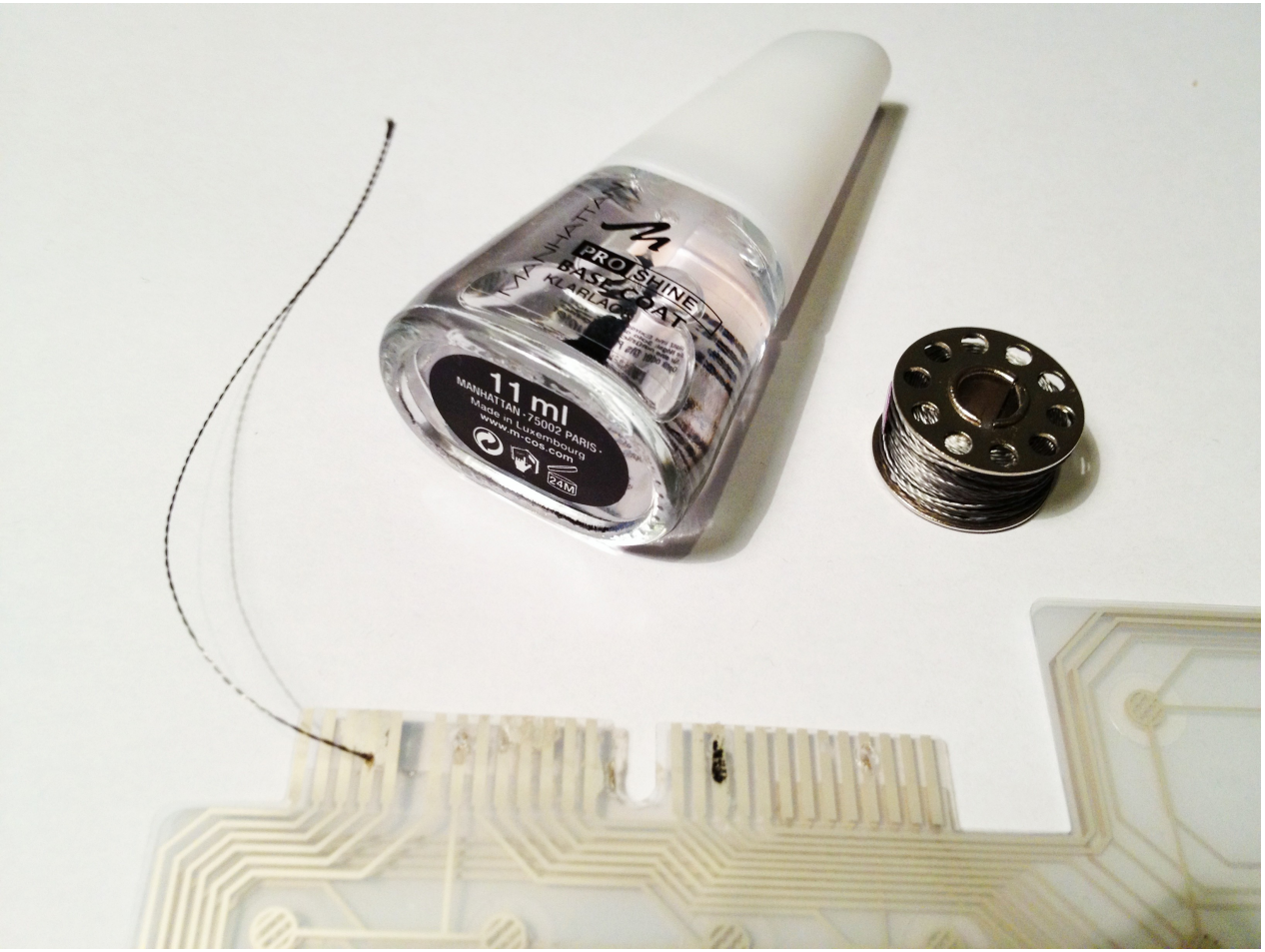
Arduino 1.5.2



What're the pitfalls?



What're the pitfalls?



Agenda

What's the problem?

What's the idea?

What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

Where do I find more?

http://calc.pw/mrmcd13

Conditions of use

You can use this OpenOffice template for your personal, educational and business presentations.

With the use of this free template you accept the following use and license conditions.

You are free:

To Share — to copy, distribute and transmit the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

No Derivative Works — You may not alter, transform, or build upon this work.

In no event shall Showeet.com be liable for any indirect, special or consequential damages arising out of or in connection with the use of the template, diagram or map.

http://www.showeet.com

http://creativecommons.org/licenses/by-nd/3.0/