calc.pw

Password Calculation
with Arduino

Who's the speaker?

Kenneth “Kenny” Newwood

E-Mail: kenneth@newwood.de

Twitter: @weizenspreu

Website: http://weizenspr.eu

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 – but they **can be reverse-engineered** or **be difficult to use**

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

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?

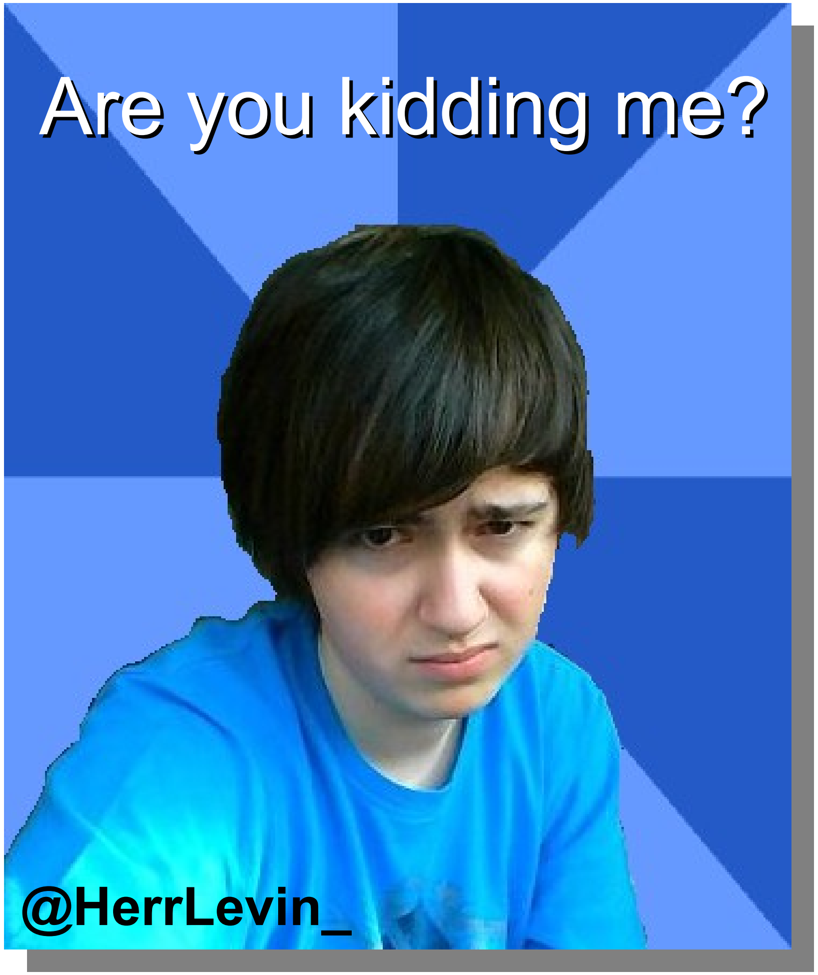
simplify password selection

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?

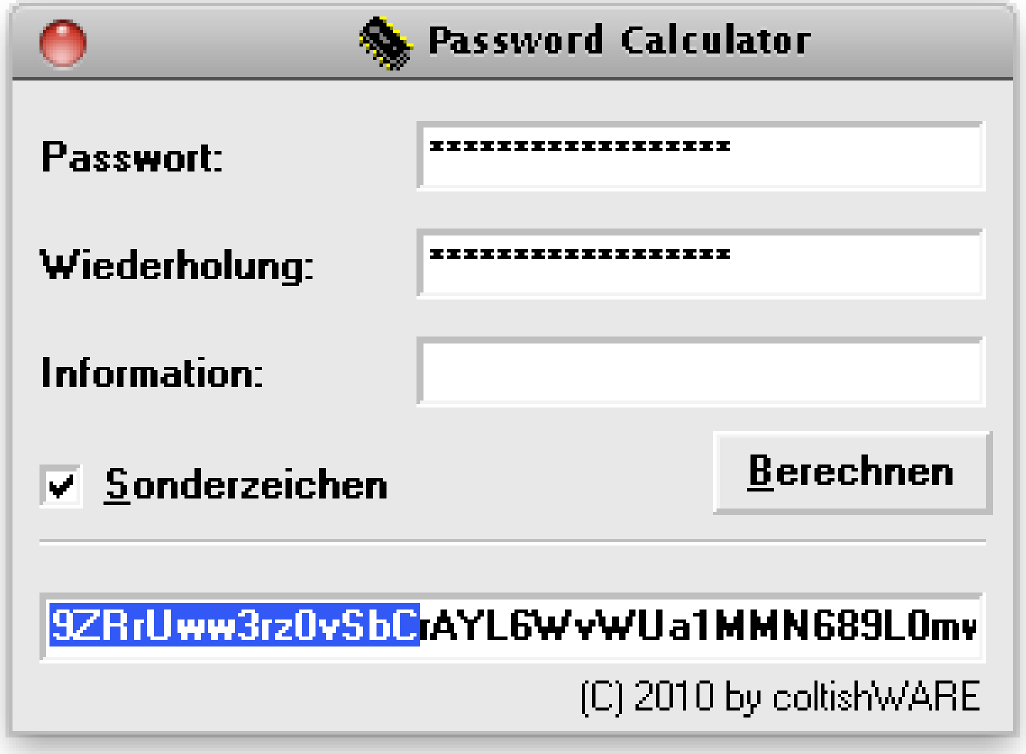
What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

What's the solution?



Passwort: ******************

Wiederholung: ******************

Information:

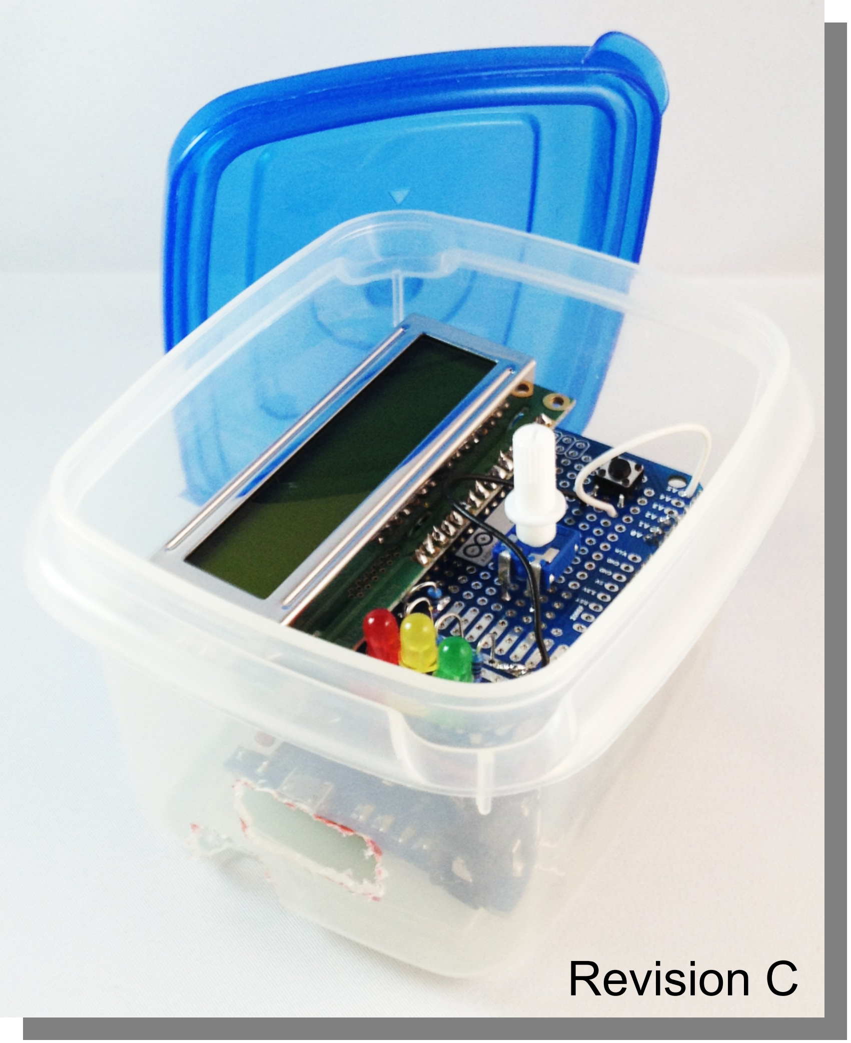
☑ Sonderzeichen

9ZRrUww3rz0vSbCrAYL6WvWUa1MMN689L0mv

(C) 2010 by coltishWARE

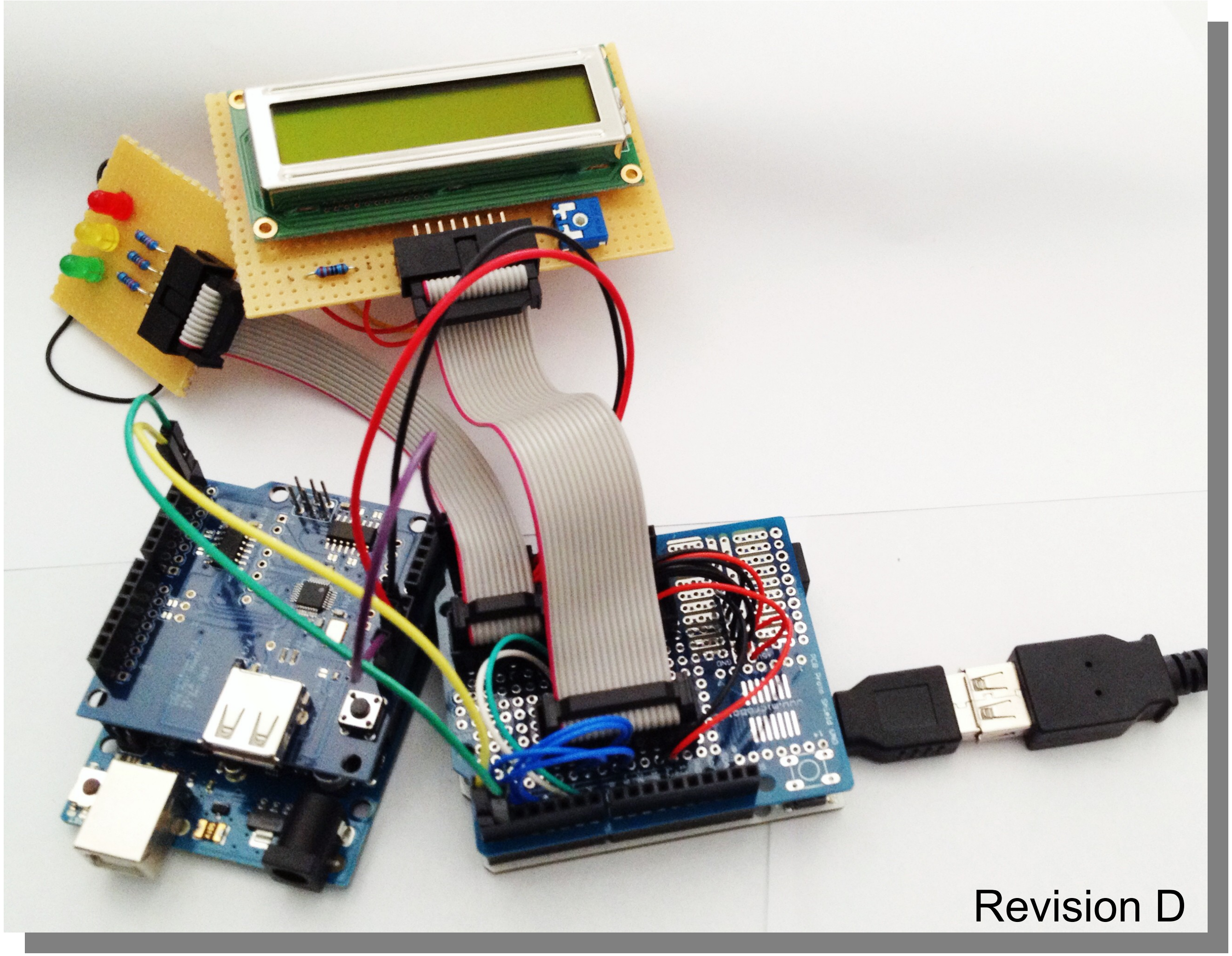
anno 2010

What's the solution?



Revision C

What's the solution?



Revision D

What's the solution?

Arduino Leonardo as output device

(keyboard emulation, LCD, LEDs)

Arduino Uno as input device

(keyboard, optionally keypad, calculation)

USB Host Shield to read keyboard input

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

Agenda

What's the problem?

What's the idea?

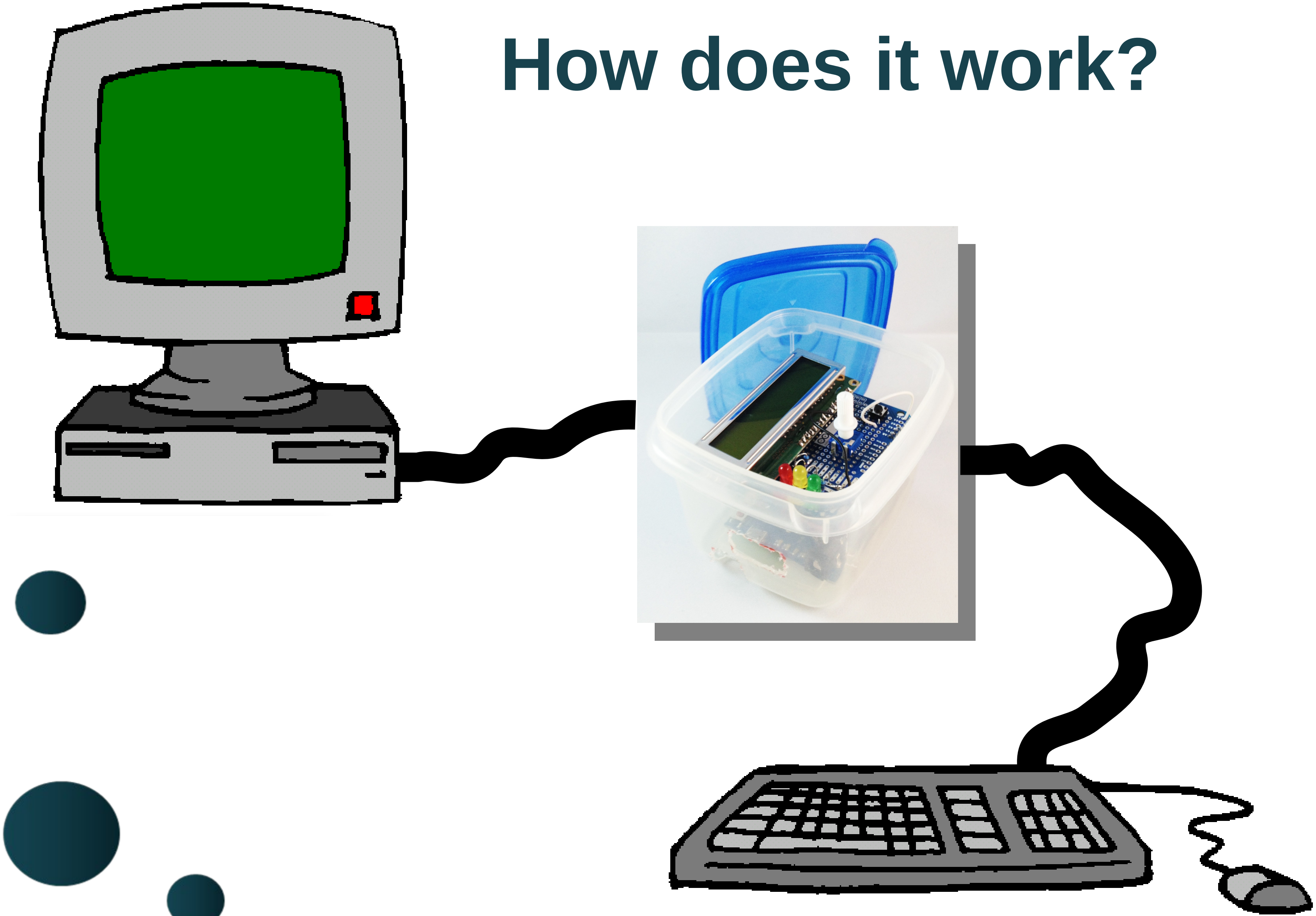
What's the solution?

How does it work?

What're the pitfalls?

Where do I find more?

How does it work?



How does it work?

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 = cleanup(crypt(hmacInfo, hmacPass))

(* simplified)

How does it work?

() = **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 = cleanup(crypt(hmacInfo, hmacPass))

(* simplified)

How does it work?

() = **SHA-1**

() = **RC4**-drop1024

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

Magic(Information, Masterpassword)* =

hmacPass = hmac(Information, Masterpassword)

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

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

Password = cleanup(crypt(hmacInfo, hmacPass))

(* simplified)

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

random access memory limitations (2.5kb)

program flash memory limitations (28kb)

mobile technology gap

What're the pitfalls?

lots of different keyboard layouts

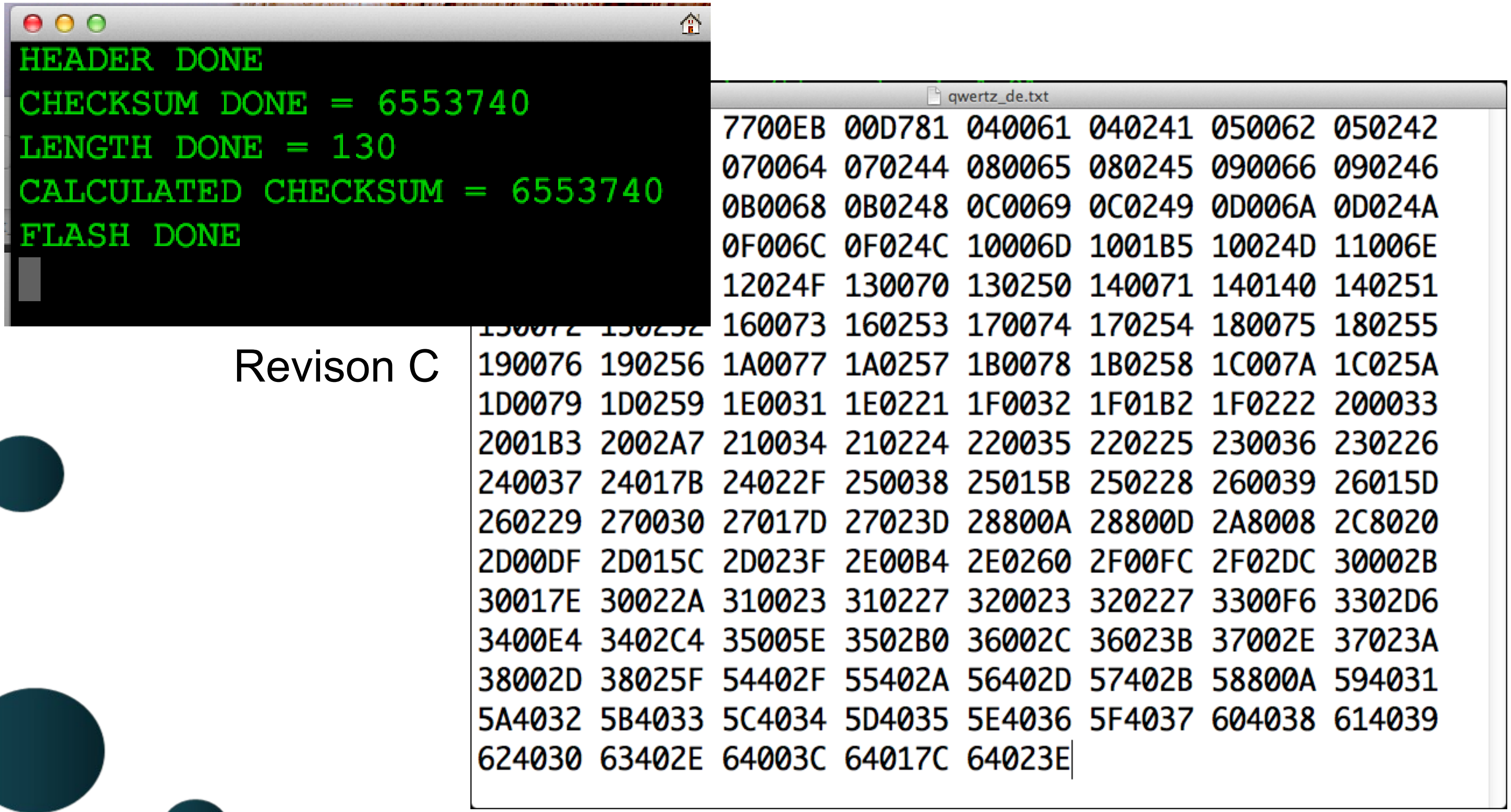
(QWERTZ DE, QWERTZ CH, QWERTZ DK, QWERTY UK, QWERTY US, AZERTY FR, AZERTY BE, Mac/Windows, etc.)

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
```

```
              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
150072 150252 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
```

Revison C

What're the pitfalls?

limited **RAM** complicates memory handling

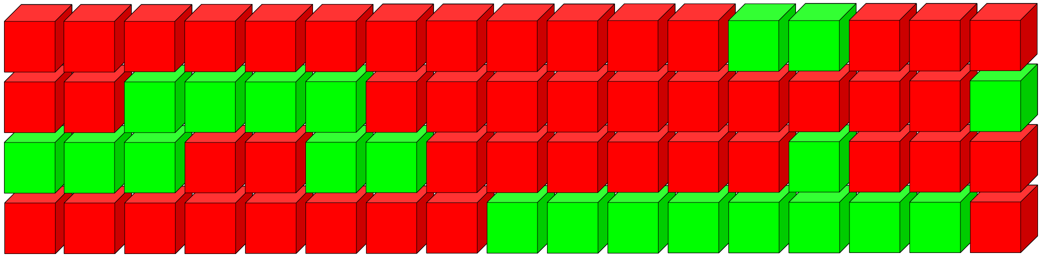
dynamic memory allocation is a **bad idea**

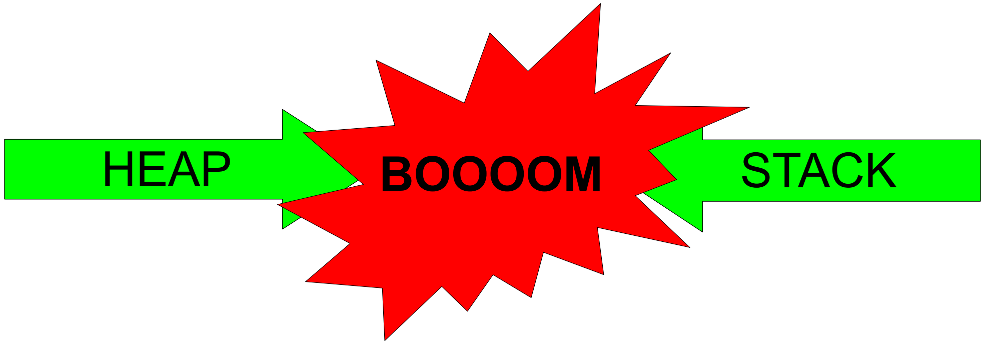
leads to fragmentation & potentially to corruption

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?





HEAP

BOOOOM

STACK

What're the pitfalls?

limited **program flash** is biggest problem

library of **USB Host Shield** grows steadily

better crypto needs more space

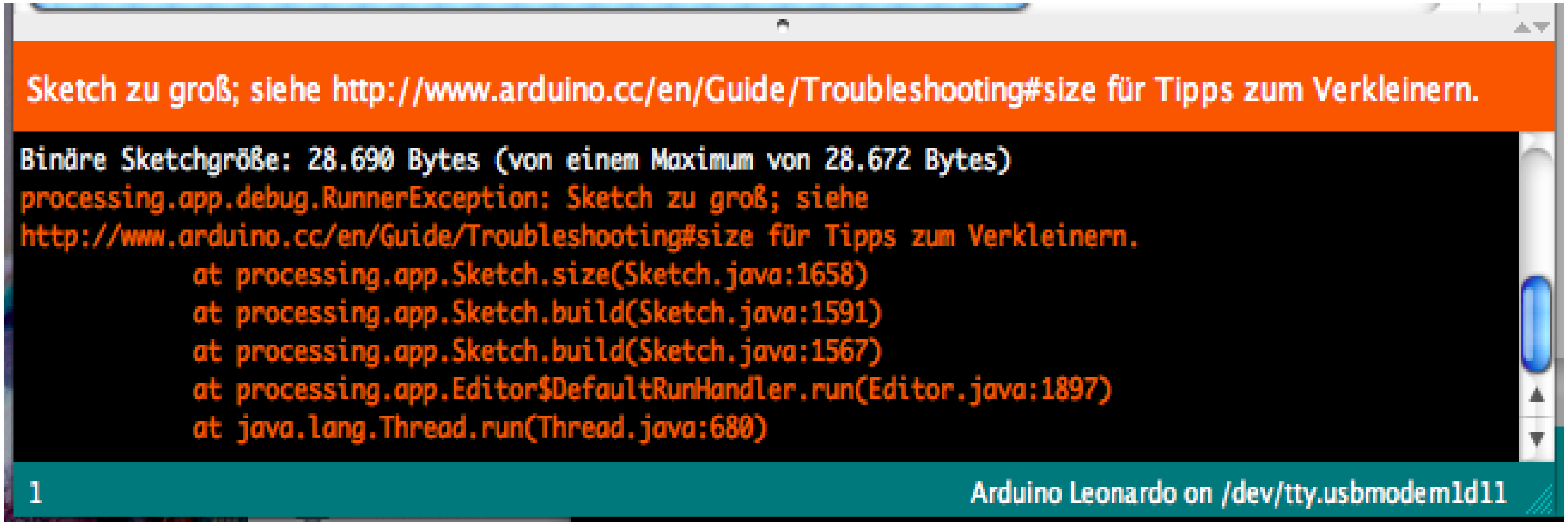
new features need more space

- type-through encryption

solution: add 2nd Arduino => divide and conquer

(benefit: core of Arduino Uno can be built for 5€)

What're the pitfalls?



Sketch zu groß; siehe http://www.arduino.cc/en/Guide/Troubleshooting#size für Tipps zum Verkleinern.

```
Binäre Sketchgröße: 28.690 Bytes (von einem Maximum von 28.672 Bytes)
processing.app.debug.RunnerException: Sketch zu groß; siehe
http://www.arduino.cc/en/Guide/Troubleshooting#size für Tipps zum Verkleinern.
	at processing.app.Sketch.size(Sketch.java:1658)
	at processing.app.Sketch.build(Sketch.java:1591)
	at processing.app.Sketch.build(Sketch.java:1567)
	at processing.app.Editor$DefaultRunHandler.run(Editor.java:1897)
	at java.lang.Thread.run(Thread.java:680)
```

Revision C

What're the pitfalls?

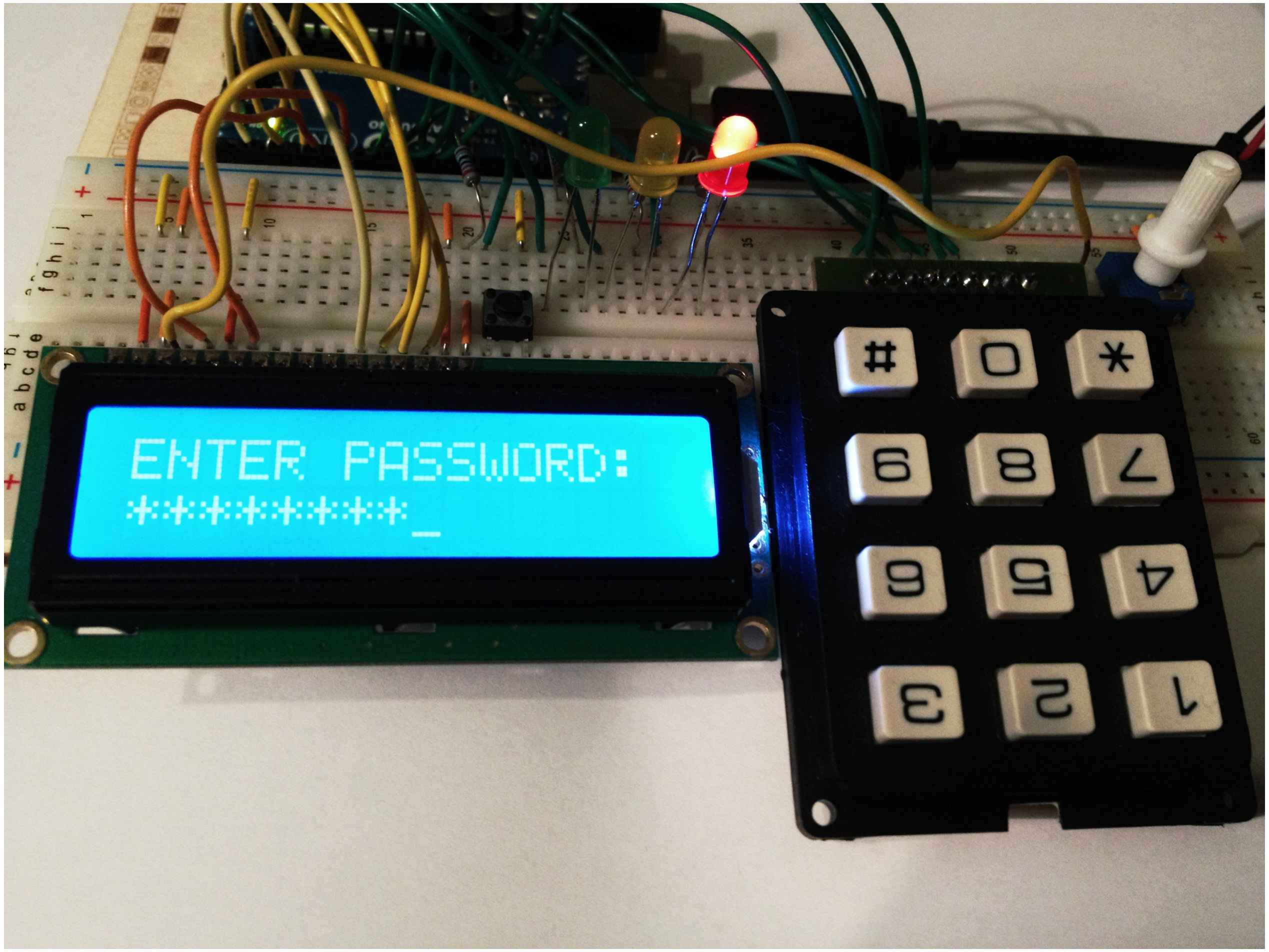
passwords need to be available on the go

USB keyboards are not an option

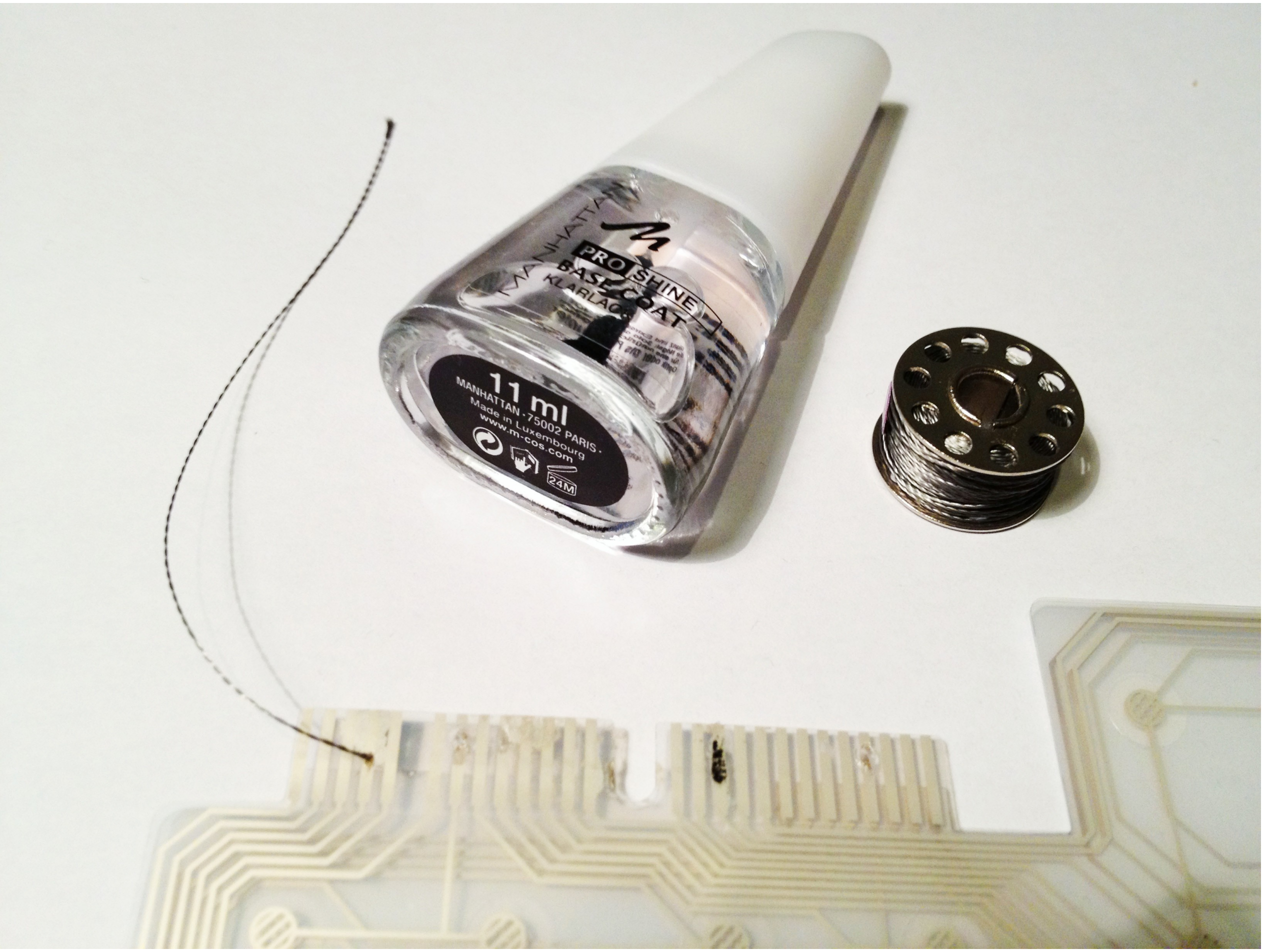
future plans:

- integrate keyboard into calc.pw
- let calc.pw act as a Bluetooth keyboard

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**/30c3

BACKUP

BACKUP

?

define length of generated password (max. 50)

!

define set of possible specials characters

#

activate check for alpha-numerics

BACKUP

SomeINFO

SomeINFO?25

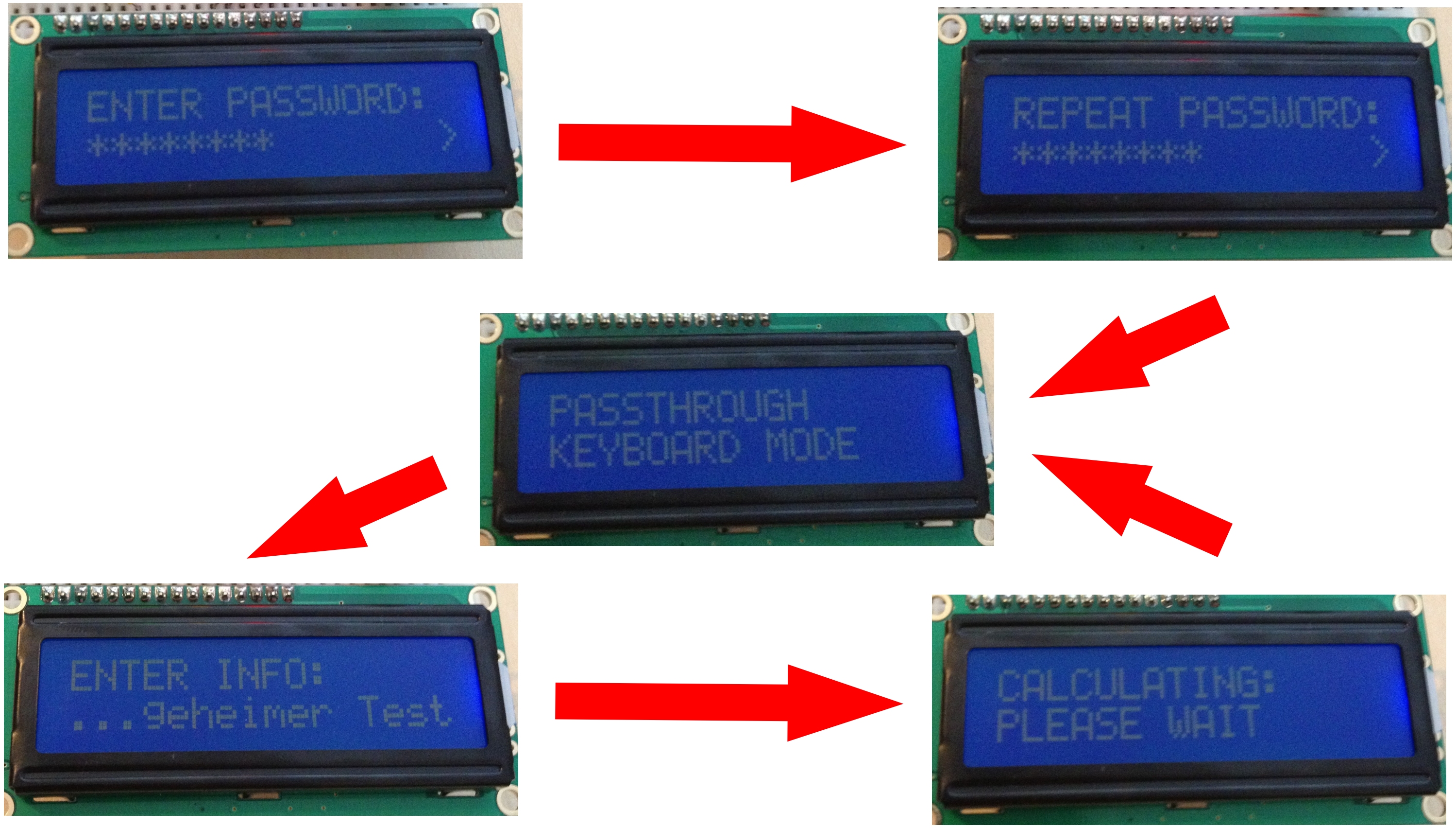
SomeINFO!+-*/

#SomeINFO

SomeINFO?25!+-*/

#SomeINFO?25!+-*/

BACKUP



```
ENTER PASSWORD:
********       >
```

```
REPEAT PASSWORD:
********       >
```

```
PASSTHROUGH
KEYBOARD MODE
```

```
ENTER INFO:
...geheimer Test
```

```
CALCULATING:
PLEASE WAIT
```

Conditions of use