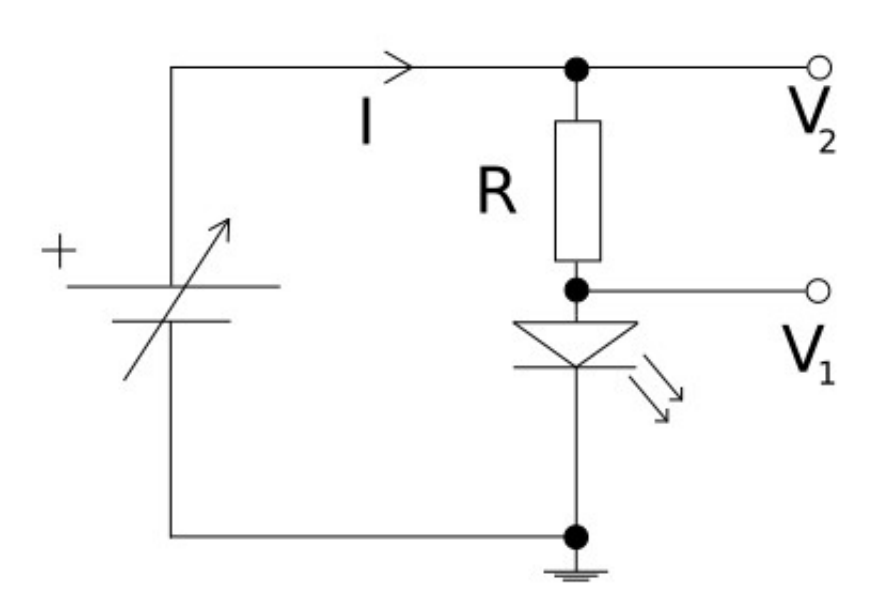
Automatizácia meraní pomocou Pythonu

“Ako naučiť prístroje rozumieť Pythonu”

Matúš Rehák
FMFI UK
matus.rehak@fmph.uniba.sk

Čo vieme pomocou Pythonu merať?

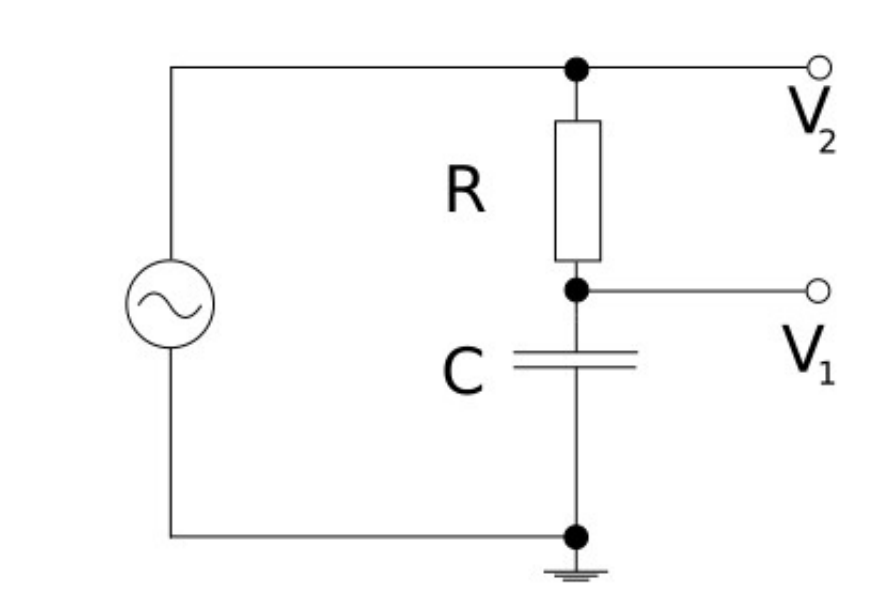
- I-V charakteristiku diódy DEMO: diode



Čo vieme pomocou Pythonu merať?

- I-V charakteristiku diódy DEMO: diode
- Frekvenčnú charakteristiku kondenzátora

  DEMO: cap



Komunikácia s prístrojom

Komunikácia s prístrojom - rozhranie

Komunikácia s prístrojom – poslanie dát

0x01

Komunikácia s prístrojom – prijatie dát

0xfa

0x01

Prekladový slovník

0x01	Zablikaj LED-kou!
0x03	Aké napätie si zmeral?
(Odpoveď) 0x00	nula voltov
(Odpoveď) 0xff	päť voltov
...	

Ako spojazdniť kúpený prístroj?

Ako spojazdniť kúpený prístroj?

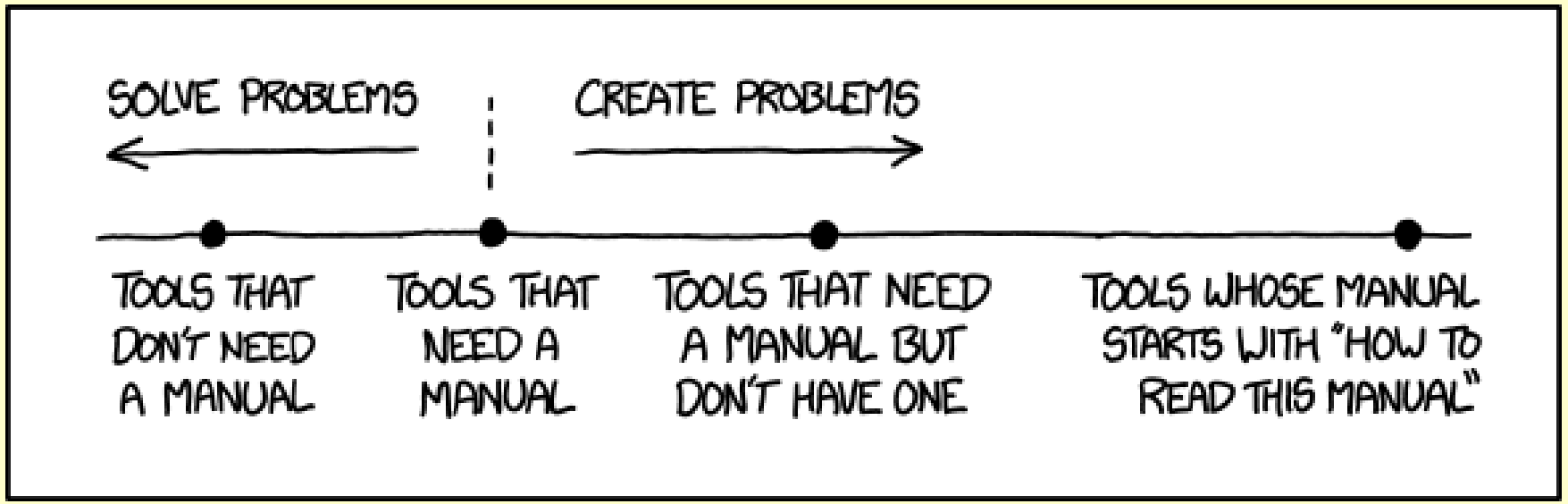
Ako spojazdniť kúpený prístroj?

Ako spojazdniť kúpený prístroj?

Ako spojazdniť kúpený prístroj?

Ako spojazdniť kúpený prístroj?

XKCD a manuály



Zdroj: http://xkcd.com/1343

Manuál, časť - programovací manuál

- Prekladový slovník

BPG_SET_TRIGGER
Select word frame trigger or clock/8.

Command (1 Byte)	Data (1 Byte)
0x02	T

T	Selected Pattern
0	word frame
1	clock/16

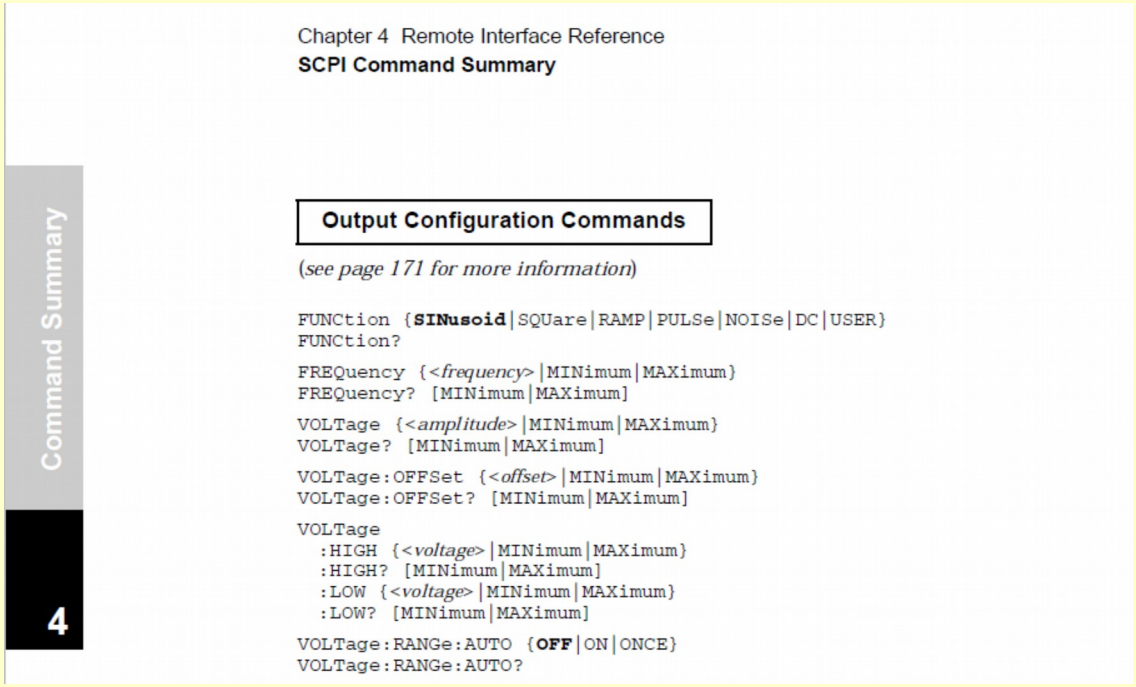
Example: Select word frame

T = 0

Command (1 Byte)	Data (1 Byte)
0x02	0x00

Manuál, časť - programovací manuál

- Prekladový slovník
- SCPI



Chapter 4 Remote Interface Reference
SCPI Command Summary

Command Summary

4

Output Configuration Commands

(*see page 171 for more information*)

```
FUNCtion {SINusoid|SQUare|RAMP|PULSe|NOISe|DC|USER}
FUNCtion?

FREQuency {<frequency>|MINimum|MAXimum}
FREQuency? [MINimum|MAXimum]

VOLTage {<amplitude>|MINimum|MAXimum}
VOLTage? [MINimum|MAXimum]

VOLTage:OFFSet {<offset>|MINimum|MAXimum}
VOLTage:OFFSet? [MINimum|MAXimum]

VOLTage
  :HIGH {<voltage>|MINimum|MAXimum}
  :HIGH? [MINimum|MAXimum]
  :LOW {<voltage>|MINimum|MAXimum}
  :LOW? [MINimum|MAXimum]

VOLTage:RANGe:AUTO {OFF|ON|ONCE}
VOLTage:RANGe:AUTO?
```

Manuál, časť - programovací manuál

- Prekladový slovník
- SCPI
- C API (.dll/.so)

3.9.30 ps6000RunBlock

```
PICO_STATUS ps6000RunBlock
(
  short              handle,
  unsigned long      noOfPreTriggerSamples,
  unsigned long      noOfPostTriggerSamples,
  unsigned long      timebase,
  short              oversample,
  long             * timeIndisposedMs,
  unsigned long      segmentIndex,
  ps6000BlockReady   lpReady,
  void             * pParameter
)
```

Pattern generator + prekladový slovník + pySerial

DEMO: serial_pat

pySerial

```
import serial


device = serial.Serial('/dev/ttyUSB0', 9600)
device.write(bytes([0x01, 0x04]))
data = device.read()    # Error
device.close()
```

Function generator + SCPI + socket

DEMO: fgen

socket

```
from socket import socket, AF_INET,SOCK_STREAM
HOST = 'google.com'     # '158.195.19.213'
PORT = 80               # 5025
message = b'GET / HTTP/1.1\n' # b'*idn?\n'


s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))
s.sendall(message)
data = s.recv(1024)
s.close()
```

Oscilloscope + C API + ctypes

DEMO: pico

ctypes

- Otvoríme shared library:

```
from ctypes import cdll #or windll
dev = cdll.LoadLibrary('/libs/mylib.so')
```

- A môžeme začať používať funkcie:

```
PICO_STATUS ps6000SetNoOfCaptures
(
  short          handle,
  unsigned long nCaptures
)
```

```
stat = dev.ps6000SetNoOfCaptures(
   ctypes.c_short(handle),
   ctypes.c_ulong(n_captures))
```

ctypes - typy

16.16.1.4. Fundamental data types

`ctypes` defines a number of primitive C compatible data types:

ctypes type	C type	Python type
`c_bool`	`_Bool`	bool (1)
`c_char`	`char`	1-character bytes object
`c_wchar`	`wchar_t`	1-character string
`c_byte`	`char`	int
`c_ubyte`	`unsigned char`	int
`c_short`	`short`	int
`c_ushort`	`unsigned short`	int
`c_int`	`int`	int
`c_uint`	`unsigned int`	int
`c_long`	`long`	int
`c_ulong`	`unsigned long`	int
`c_longlong`	`__int64` or `long long`	int
`c_ulonglong`	`unsigned __int64` or `unsigned long long`	int
`c_size_t`	`size_t`	int
`c_ssize_t`	`ssize_t` or `Py_ssize_t`	int
`c_float`	`float`	float
`c_double`	`double`	float
`c_longdouble`	`long double`	float
`c_char_p`	`char *` (NUL terminated)	bytes object or `None`
`c_wchar_p`	`wchar_t *` (NUL terminated)	string or `None`
`c_void_p`	`void *`	int or `None`

https://docs.python.org/3/library/ctypes.html

ctypes – Čo s pointermi?

- ctypes má “byref”!

```
PICO_STATUS ps6000OpenUnit
(
  short * handle,
  char  * serial
)
```

```
handle = ctypes.c_short()    #c_short(0)
serial_n = create_string_buffer(b'AY166/090')

stat = dev.ps6000OpenUnit(
   ctypes.byref(handle),
   ctypes.byref(serial))
print(handle)                #c_short(6154)
```

ctypes – Je to všetko?

- ctypes umožňuje vytváranie polí

```
FiveIntArrayType = ctypes.c_int * 5
my_array = FiveIntArrayType(0, 1, 2, 3, 4)
```

- ctypes umožňuje vytváranie štruktúr

```
from ctypes import Structure, c_int, c_float
class AB_struct(Structure):
   _fields_ = [('a', c_int), ('b', c_float)]
my_struct = AB_struct()
my_struct.b = c_float(1.5)
```

- atď.

Sumarizácia + záverečné poznámky

Ak prístroj rozumie byte-om alebo SCPI:

- pySerial
- socket

Ak prístroj rozumie C-čku:

- ctypes

Čo sa oplatí ešte pozrieť:

- pyVISA
- QTLab

Sumarizácia + záverečné poznámky

Čo sa dá ešte spraviť s pythonom a function generatorom?

DEMO: p_a_m

ĎAKUJEM!


SRS
STANFORD RESEARCH SYSTEMS
SIM900 MAINFRAME
CAU
P4-
2016