01010010

Speak Binary to Me

Troels Brødsgaard
@trarbr [github, twitter, genserver.social]

inilab.dk

"Smart" stuff

- Remote terminal units (RTU)
- Programmable logic controllers (PLC)
- Solar inverters
- Heatpumps
- Electricity meters
- Thermostats, window blinds, door contacts
- Various sensors and actuators

The BEAM advantage

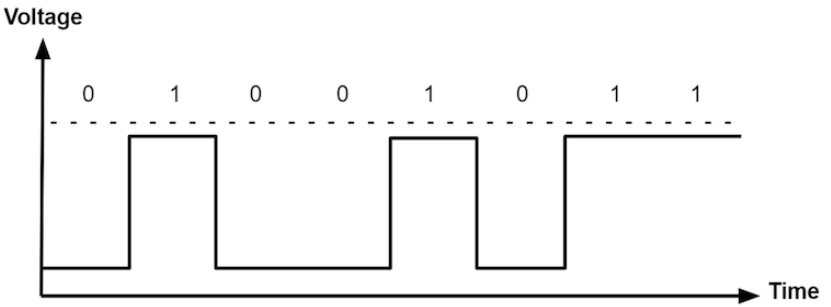
- Fault tolerance
- State machines
- Concurrency

The BEAM advantage

- Fault tolerance
- State machines
- Concurrency
- **Bit syntax**

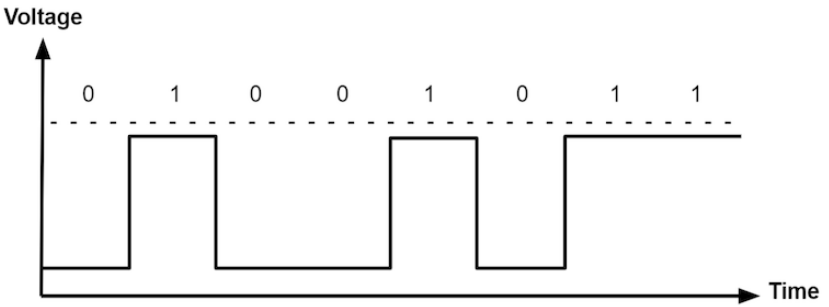
BINARIES

Computers



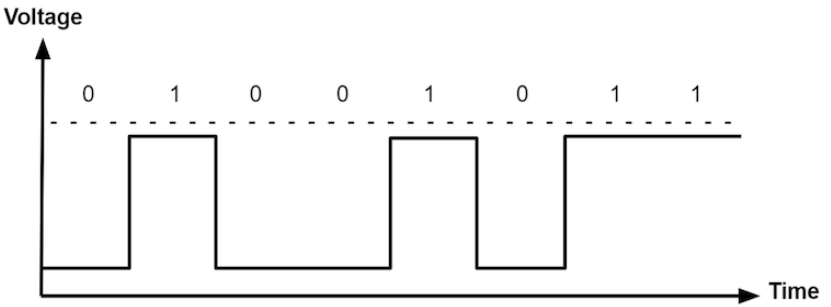
Binaries and notations

```
0b01001011 <=>   75    <=> 0x4B
  Binary   <=> Decimal <=> Hex
```



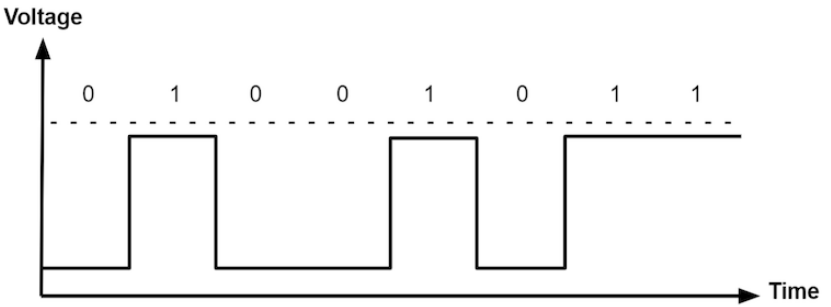
Binaries and notations

```
0b01001011 <=>   75    <=> 0x4B
  Binary   <=> Decimal <=> Hex
```



Binaries and notations

```
0b01001011 <=>   75    <=> 0x4B
  Binary   <=> Decimal <=> Hex
```



IPv4 address

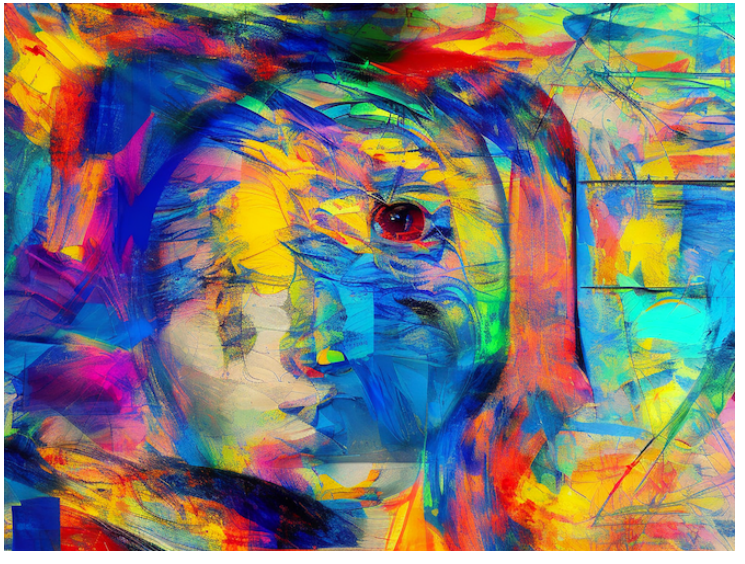
`192.168.1.10`

`0b11000000 0b10101000 0b00000001 0b00001010`

MAC address

```
AD:04:5E:00:53:AF

0b10101101 0b00000100 0b01011110
0b00000000 0b01010011 0b10101111
```



BIT SYNTAX

Encoding a binary

```
my_binary = ??????????
```

Encoding a binary

```
my_binary = <<10, 20, 30>>
```

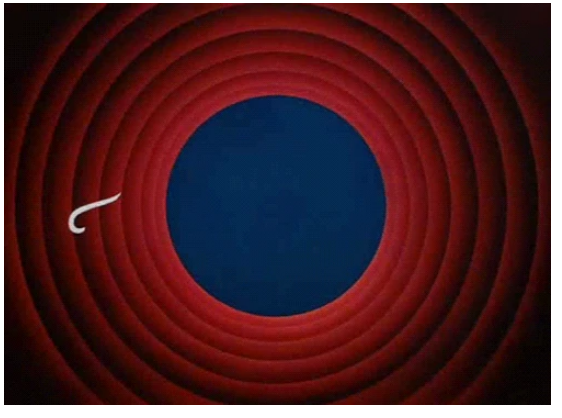
Decoding a binary

```
# my_binary = <<10, 20, 30>>
?????????? = my_binary
# a = 10, b = 20, c = 30
```

Decoding a binary

```
# my_binary = <<10, 20, 30>>
<<a, b, c>> = my_binary
# a = 10, b = 20, c = 30
```

Are we done?



Encoding with modifiers

```
my_binary = <<
  10::integer-unsigned-size(8)-unit(1)-big,
  20::integer-unsigned-8-unit(1)-big,
  30
>>
```

Modifiers: **type**, **sign**, **size**, **unit** and **endianness**.

Decoding with modifiers

```
<<
  a::integer-unsigned-size(8)-unit(1)-big,
  b::integer-unsigned-8-unit(1)-big,
  c
>> = my_binary
```

Protocol programming in Erlang using binaries

Claes Wikström Tony Rogvall*
Computer Science Laboratory
Ericsson Telecom AB

29 Sep 1998

Back to endianness

```
41665 == 0xA2C1
```

Back to endianness

```
41665 == 0xA2C1

                MSB  LSB
0xA2C1 <-- 0xA2 0xC1 --> 0xC1A2

Big-endian              Little-endian
```

IEN 137

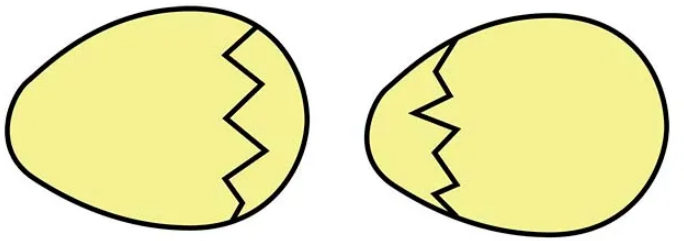
Danny Cohen
U S C/I S I
1 April 1980

ON HOLY WARS AND A PLEA FOR PEACE

INTRODUCTION

This is an attempt to stop a war. I hope it is not too late and that somehow, magically perhaps, peace will prevail again.

Gulliver's Travels

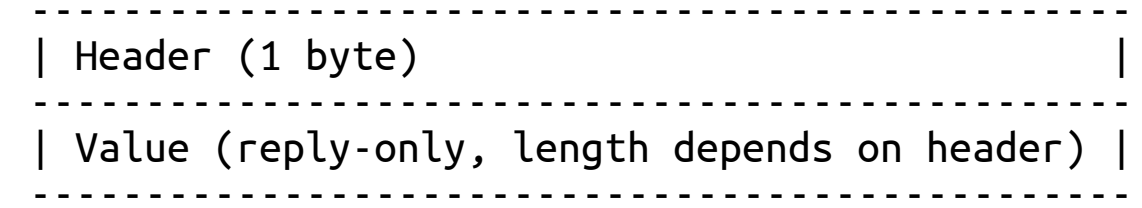


EXAMPLES

TBox

Has name, measures temperature

TBox message format

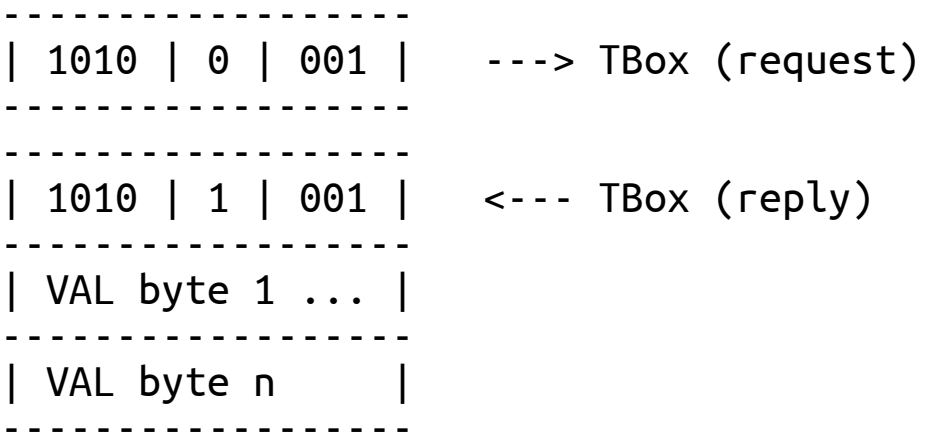


TBox header

```
-----------------------------------------------
| MAGIC (4 bits) | DIR (1 bit) | ATT (3 bits) |
-----------------------------------------------
```

- **MAGIC**: Constant (`0xA` / `0b1010`)
- **DIR**: Request (`0x0` / `0b0`) or Reply (`0x1` / `0b1`)
- **ATT**: NAME (`0x0` / `0b000`), TEMP (`0x1` / `0b001`)

TBox request/reply



TBox VAL name

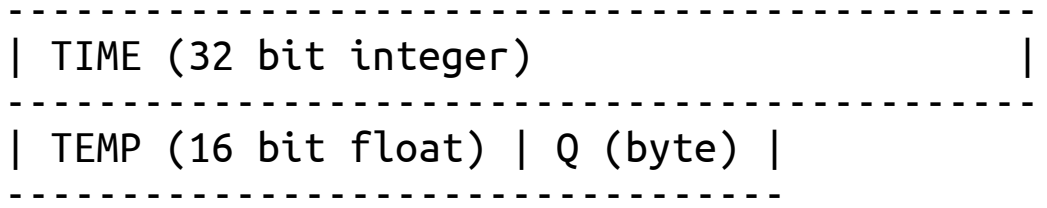
```
-------------------------
| NAME (12 ASCII bytes) |
-------------------------
```

TBox name example

```
"FOSDEM"

-----------------------------------------
| 01000110 | 01001111 | 01010011 | 01000100 |
-----------------------------------------
| 01000101 | 01001101 | 00000000 | 00000000 |
-----------------------------------------
| 00000000 | 00000000 | 00000000 | 00000000 |
-----------------------------------------
```

TBox VAL temperature



Numbers are little-endian!

TBox temperature example

```
-------------------------------------------
| 11010101 | 01111101 | 11011001 | 01100011 |
-------------------------------------------
| 00000100 | 01010000 | 00000011 |
-----------------------------------
```

Get TBox name

```
request = <<0xA::4, 0::1, 0::3>>
```

Get TBox name

```
request = <<0xA::4, 0::1, 0::3>>
```

```
<<
  0xA::4, 1::1, 0::3,
  name::bytes-12
>> = reply
```

Get TBox temperature

```
request = <<0xA::4, 0::1, 1::3>>
```

Get TBox temperature

```
request = <<0xA::4, 0::1, 1::3>>
```

```
<<
  0xA::4, 1::1, 1::3,
  timestamp::32-little,
  temperature::float-16-little,
  _::6, clock_error::1, temperature_error::1
>> = reply
```

So much to cover

- Streaming
- Generators
- Performance tuning
- Other tools (Wireshark, tcpdump, btmon...)

Check out Protohackers

- protohackers.com
- Andrea Leopardi's live stream on YouTube

Thank you

Troels Brødsgaard
@trarbr [github, twitter, genserver.social]