Lua multi VM system for home automation

Lua Workshop 2012

Filip Zawadiak, DOMIQ Sp. z o.o.
fzawadiak@domiq.pl

About DOMIQ

- Develops home automation integration & user interface modules
- Main module + extension modules for particular protocols
- Currently: LCN, MODBUS, DMX, SATEL, DALI, SONOS
- Soon: KNX, BACnet and LON
- Fairly low volume product and highly customizable
- Frequent software releases, small team

About Building/Home Automation

- Soft-realtime, users complain about delays above 500ms
- Multiple subsystems, multiple protocols, all relatively slow, 9600bps
- Typical cost for home installation 10-30kUSD
- Needs to be easy to program & configure – usually performed by electricians
- Extensibility is extremely important – lots of “weird” ideas from customers
- Some unusual installations: large office buildings, hospital, church

Lua usage in DOMIQ

- **Base** – Dedicated Lua based multi VM OS and custom hardware
- **Display** – Based on Linux, with custom UI library built on Microwindows
- **Server** – Message routing hub for customers

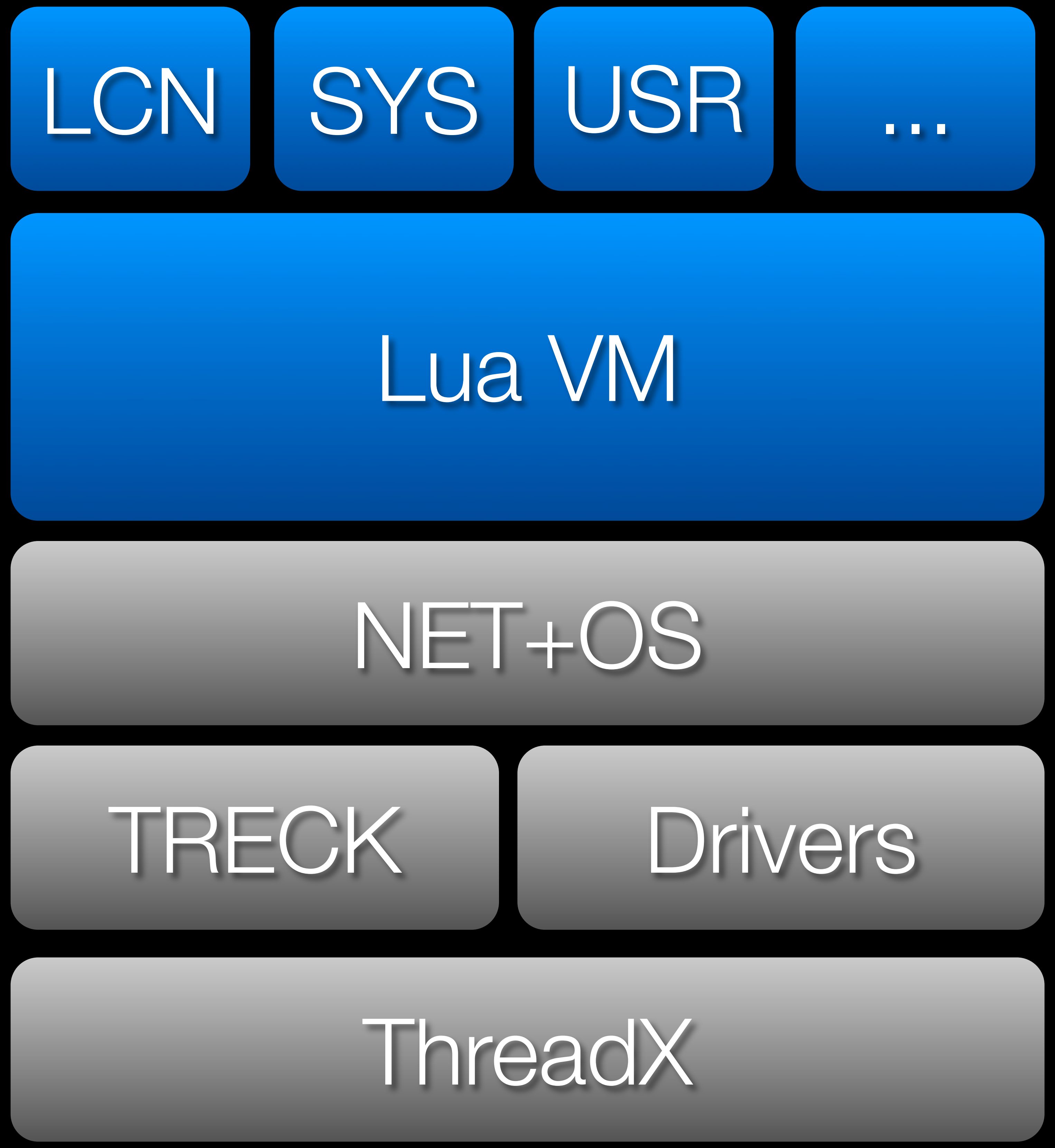
- Custom programming in Lua exposed for end customers
- Protocol protyping, encoding design etc
- It’s addictive

DOMIQ/Base

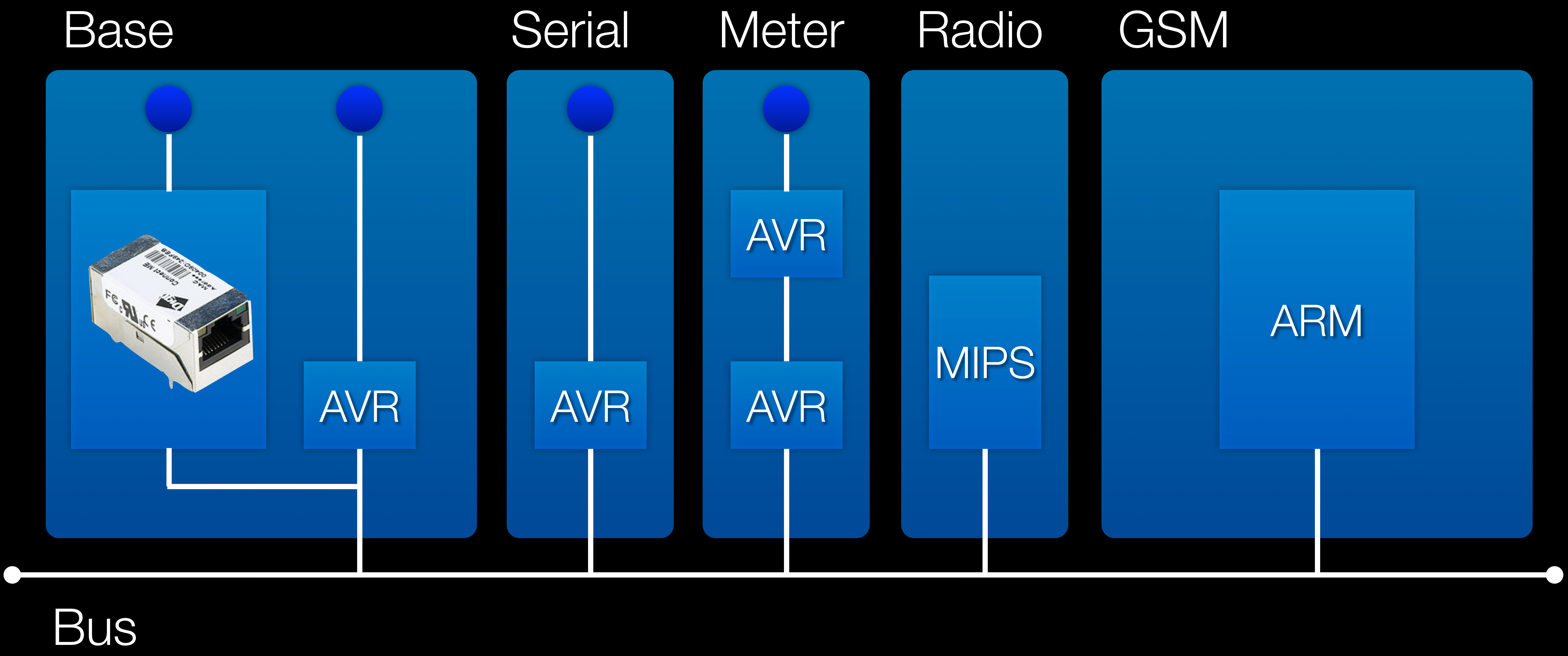
- "Server" module
- 8MB RAM, 4MB FLASH
- 75MHz ARM
- Ethernet connection
- Built-in LCN interface
- Custom software stack based on NET+OS from DIGI

Software Architecture

- Based on NET+OS from DIGI
  - TRECK IP Stack
  - ThreadX RTOS
- Unique platform
  - Lua Virtual Machines
  - Publish/Subscribe Channels
  - Extremely Memory Efficient



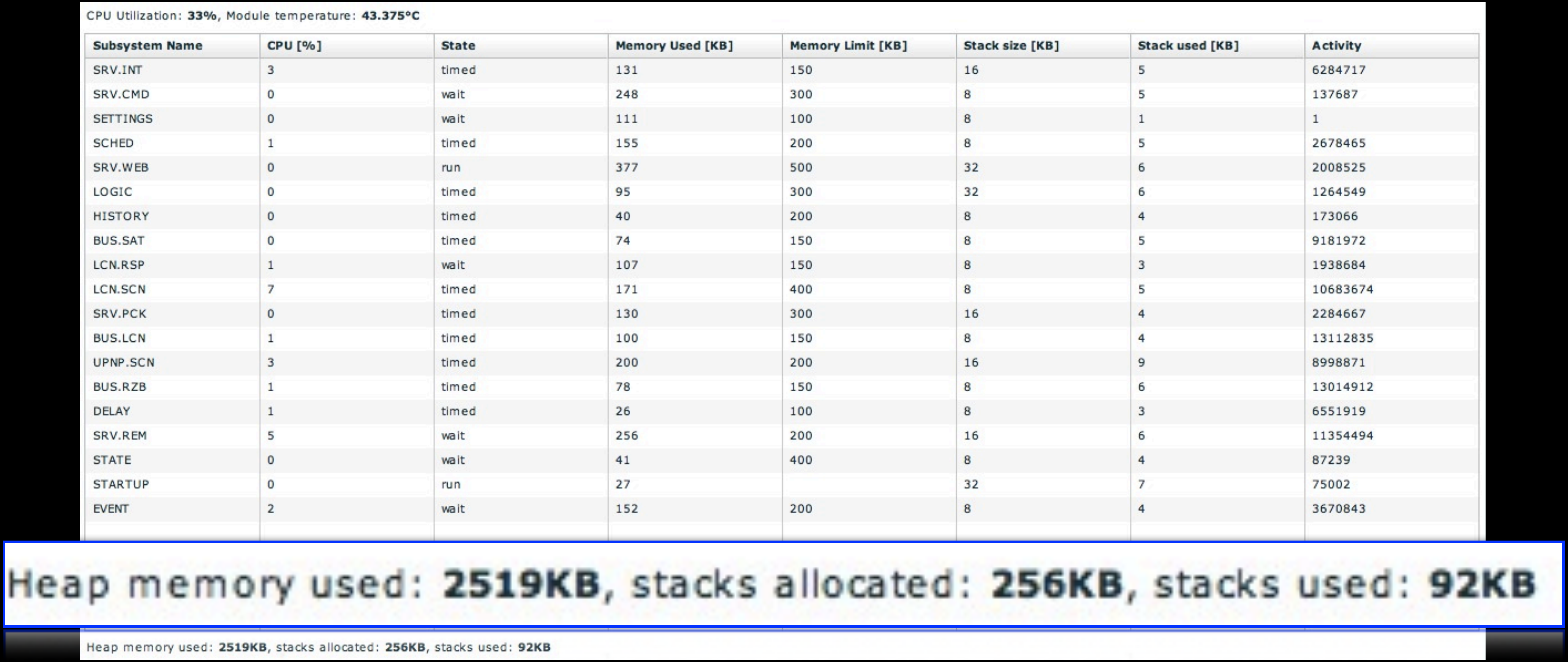
System Architecture



Multi VM System

- Multiple Lua VMs – each with local state & global lock
- Three main communication mechanisms
  - Asynchronous Lua code message
  - Synchronous Lua code message
  - Synchronous inter-VM method call
  - Publish-Subscribe Channels

Memory Efficient Software



CPU Utilization: **33%**, Module temperature: **43.375°C**

Subsystem Name	CPU [%]	State	Memory Used [KB]	Memory Limit [KB]	Stack size [KB]	Stack used [KB]	Activity
SRV.INT	3	timed	131	150	16	5	6284717
SRV.CMD	0	wait	248	300	8	5	137687
SETTINGS	0	wait	111	100	8	1	1
SCHED	1	timed	155	200	8	5	2678465
SRV.WEB	0	run	377	500	32	6	2008525
LOGIC	0	timed	95	300	32	6	1264549
HISTORY	0	timed	40	200	8	4	173066
BUS.SAT	0	timed	74	150	8	5	9181972
LCN.RSP	1	wait	107	150	8	3	1938684
LCN.SCN	7	timed	171	400	8	5	10683674
SRV.PCK	0	timed	130	300	16	4	2284667
BUS.LCN	1	timed	100	150	8	4	13112835
UPNP.SCN	3	timed	200	200	16	9	8998871
BUS.RZB	1	timed	78	150	8	6	13014912
DELAY	1	timed	26	100	8	3	6551919
SRV.REM	5	wait	256	200	16	6	11354494
STATE	0	wait	41	400	8	4	87239
STARTUP	0	run	27		32	7	75002
EVENT	2	wait	152	200	8	4	3670843

Heap memory used: **2519KB**, stacks allocated: **256KB**, stacks used: **92KB**

Heap memory used: **2519KB**, stacks allocated: **256KB**, stacks used: **92KB**

VM Management

- Start new VM
  `vm.start(name,quota,priority,stack)`
- Stop VM
  `vm.stop(name)`
- Get list of VMs
  `vm.list()`

Inter VM Calls

- Enqueue Lua code to execute in named VM
  `vm.execute(name,code)`
- Synchronously execute Lua code in named VM, copy results
  `vm.call(name,code)`
- Synchronously execute Lua method by path in VM, copy params & results
  `vm.qcall(name,path,...)`

Publish-Subscribe

- Call handler(channel,data) any time message is posted
  `vm.subscribe(prefix,handler)`
- Cancel subscription
  `vm.unsubscribe(prefix)`
- Post message to channel with data
  `vm.post(channel,data)`

Timers

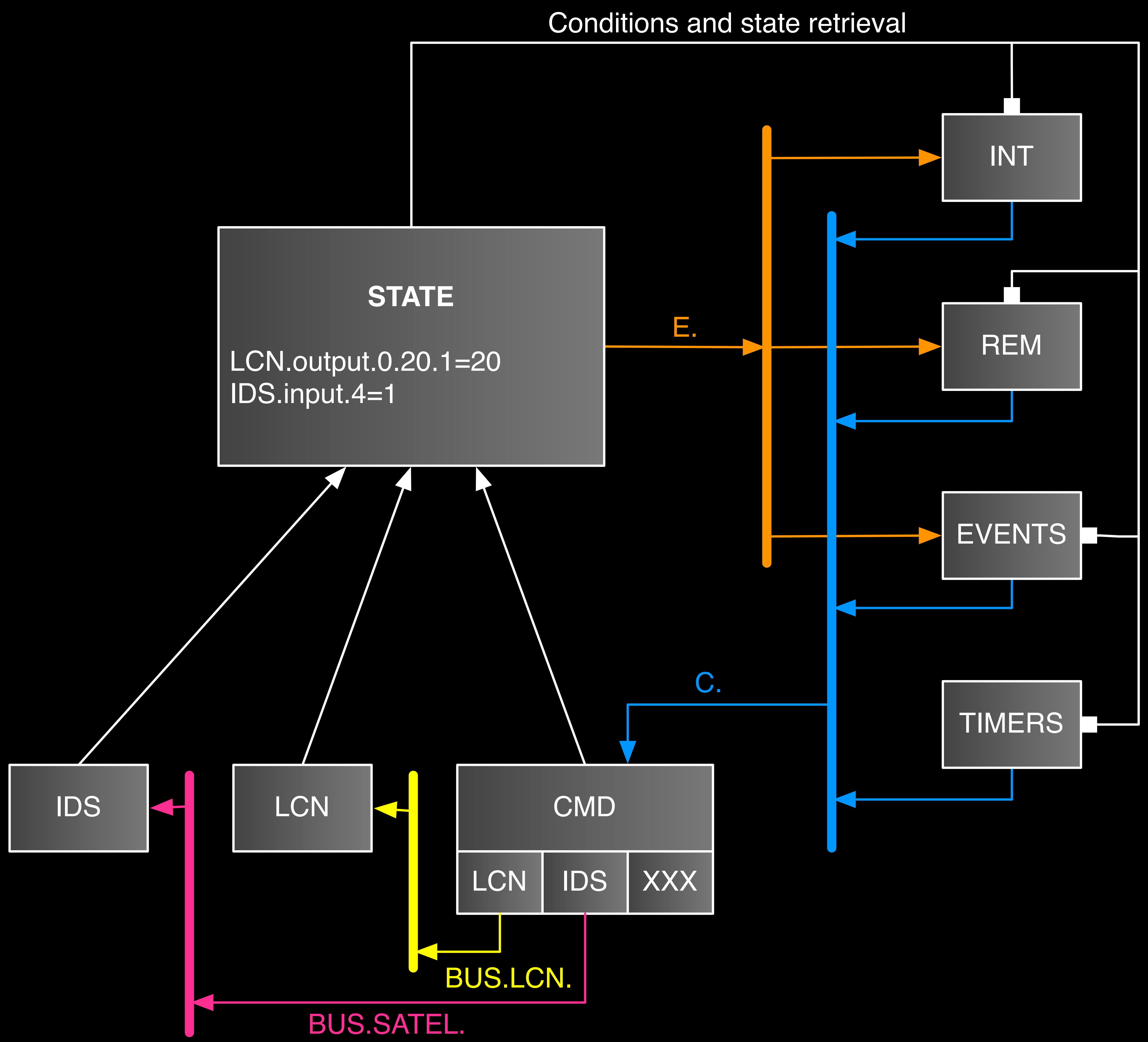
- Old style, execute once
  `vm.timer(timeout,handler)`
- New style
  `vm.timer(name,timeout,repeat,handler)`

```
NAME               STR  ALL  LMT GC  STA  ENQ  CPU  STK  OVH
BOOT               161   15    0     run    0   0%    3    1
STATE              683   71  400     tim    0   0%    5    6
SETTINGS           823  127  150  F  sus    0   0%    1   11
BUS.LCN            660  111  150     tim    0   3%    4    9
BUS.SAT            572   71  150     tim    0   2%    5    6
HISTORY            299   37  200     tim    0   0%    4    3
SRV.REM           1029  200  300     sus    0   6%    6   17
SRV.INT            603  138  200     run    0   2%    5   11
LCN.SCN            943  188  400     tim    0   6%    5   14
LCN.RSP            707  123  150     sus    0   0%    4   10
SRV.CMD           1157  226  300     sus    0   2%    5   20
DELAY              283   33  100     tim    0   0%    3    3
LOGIC              530   85  300     tim    0   0%    5    8
SCHED              884  172  300     tim    0   7%    5   16
EVENT              985  210  300     sus    0  11%    4   22
SRV.PCK            665  130  150     sus    0   2%    4   13
SRV.CAM            521   87  150     run    0   0%    4    9
SRV.WEB           1865  366  500     sus    0   0%    5   34
SRV.DEA            907  164  250     run    0   3%    9   14
SRV.MOD            583  142  250     tim    0   7%    4    8
SRV.UAV           1315  277  300     run    0   2%    9   26

TOTALS: strings=16175 alloc=2701 limit=5000 overhead=261 heap=4096 free=1395
```

Data Flow

- Only VMs involved in state keeping are visible
- White lines are direct VM calls
- **Events**
- **Commands**



Priority Auto Tuning

- Each VM is started with configurable priority
- Priority can be raised when message queue becomes full

Memory Allocations

- TLFS memory allocator – less fragmentation
- Configurable “quotas” for heap usage and variable GC speed
- In case of allocation failure: force all VMs to do full GC & try later

Bytecode and PAK files

- Lua bytecode much larger than source code
- Special file format – compressed data, lockable files, integrity checks
- Automatic decompression & caching

- Tools for endianness changes: ChunkSpy.lua and eLua cross patch

Lua Usage

- Standard Libraries
  - bitop
  - lfs
  - xavante
  - luasoap
  - luasocket
  - copas
  - struct
- Custom Libraries
  - vm
  - diq
  - binary
  - packet
  - aes
  - ecc
  - pak

Binary

- Binary arrays of predefined types
- Used to store temperature history and large flag sets
- Indexed access + ability to roll data

Packet

- Not used yet...
- Will replace strings as main data type for protocol implementation
- Struct like access by offset and binary types
- Prepend/Append with optional preallocated space
- Should also be used for cross VM messaging

Live Demo – VM interaction

Plans for future

- Update to current Lua version
- Possibly use LuaJIT
- Callback based TCP connectivity
- Integrate SQLite3
- New generation of hardware
- Possibly customized hardware for Lua

Lua – excellent choice

- We were able to develop quite large software stack in very short time
- It runs nicely on very small hardware
- Multi-VM design makes concurrent programming easy
- “Relatively easy” to learn for electricians :-)

Questions?

See also **www.domiq.eu** for more information