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 prototyping, 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
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

<table>
<thead>
<tr>
<th>Subsystem Name</th>
<th>CPU [%]</th>
<th>State</th>
<th>Memory Used [KB]</th>
<th>Memory Limit [KB]</th>
<th>Stack size [KB]</th>
<th>Stack used [KB]</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRV.INT</td>
<td>3</td>
<td>timed</td>
<td>131</td>
<td>150</td>
<td>16</td>
<td>5</td>
<td>6284717</td>
</tr>
<tr>
<td>SRV.CMD</td>
<td>0</td>
<td>wait</td>
<td>248</td>
<td>300</td>
<td>8</td>
<td>5</td>
<td>137687</td>
</tr>
<tr>
<td>SETTINGS</td>
<td>0</td>
<td>wait</td>
<td>111</td>
<td>100</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SCHED</td>
<td>1</td>
<td>timed</td>
<td>155</td>
<td>200</td>
<td>8</td>
<td>5</td>
<td>2678465</td>
</tr>
<tr>
<td>SRV.WEB</td>
<td>0</td>
<td>run</td>
<td>277</td>
<td>500</td>
<td>32</td>
<td>6</td>
<td>2008525</td>
</tr>
<tr>
<td>LOGIC</td>
<td>0</td>
<td>timed</td>
<td>95</td>
<td>300</td>
<td>32</td>
<td>6</td>
<td>1264549</td>
</tr>
<tr>
<td>HISTORY</td>
<td>0</td>
<td>timed</td>
<td>40</td>
<td>200</td>
<td>8</td>
<td>4</td>
<td>173066</td>
</tr>
<tr>
<td>BUS.SAT</td>
<td>0</td>
<td>timed</td>
<td>74</td>
<td>150</td>
<td>8</td>
<td>5</td>
<td>9181972</td>
</tr>
<tr>
<td>LCR.RSP</td>
<td>1</td>
<td>wait</td>
<td>107</td>
<td>150</td>
<td>8</td>
<td>3</td>
<td>193666</td>
</tr>
<tr>
<td>LCR.SCN</td>
<td>7</td>
<td>timed</td>
<td>171</td>
<td>400</td>
<td>8</td>
<td>5</td>
<td>10683674</td>
</tr>
<tr>
<td>SRV.PCK</td>
<td>0</td>
<td>timed</td>
<td>130</td>
<td>300</td>
<td>16</td>
<td>4</td>
<td>2284667</td>
</tr>
<tr>
<td>BUS.LCN</td>
<td>1</td>
<td>timed</td>
<td>100</td>
<td>150</td>
<td>8</td>
<td>4</td>
<td>13112835</td>
</tr>
<tr>
<td>UPNP.SCN</td>
<td>3</td>
<td>timed</td>
<td>200</td>
<td>200</td>
<td>16</td>
<td>9</td>
<td>8998871</td>
</tr>
<tr>
<td>BUS.RZB</td>
<td>1</td>
<td>timed</td>
<td>78</td>
<td>150</td>
<td>8</td>
<td>6</td>
<td>13014912</td>
</tr>
<tr>
<td>DELAY</td>
<td>1</td>
<td>timed</td>
<td>26</td>
<td>100</td>
<td>8</td>
<td>3</td>
<td>65551919</td>
</tr>
<tr>
<td>SRV.REM</td>
<td>5</td>
<td>wait</td>
<td>256</td>
<td>200</td>
<td>16</td>
<td>6</td>
<td>11354694</td>
</tr>
<tr>
<td>STATE</td>
<td>0</td>
<td>wait</td>
<td>41</td>
<td>400</td>
<td>8</td>
<td>4</td>
<td>87239</td>
</tr>
<tr>
<td>STARTUP</td>
<td>0</td>
<td>run</td>
<td>27</td>
<td>32</td>
<td>8</td>
<td>4</td>
<td>75052</td>
</tr>
</tbody>
</table>

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

- Start new VM
  
  ```python
  vm.start(name, quota, priority, stack)
  ```

- Stop VM
  
  ```python
  vm.stop(name)
  ```

- Get list of VMs
  
  ```python
  vm.list()
  ```
Inter VM Calls

- Enqueue Lua code to execute in named VM
  \[\text{vm.execute}(\text{name}, \text{code})\]

- Synchronously execute Lua code in named VM, copy results
  \[\text{vm.call}(\text{name}, \text{code})\]

- Synchronously execute Lua method by path in VM, copy params & results
  \[\text{vm.qcall}(\text{name}, \text{path}, \ldots)\]
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
  \[\text{vm.timer(timeout,handler)}\]

- New style
  \[\text{vm.timer(name,timeout,repeat,handler)}\]
<table>
<thead>
<tr>
<th>NAME</th>
<th>STR</th>
<th>ALL</th>
<th>LMT</th>
<th>GC</th>
<th>STA</th>
<th>ENQ</th>
<th>CPU</th>
<th>STK</th>
<th>OVH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT</td>
<td>161</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>run</td>
<td>0</td>
<td>0%</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>STATE</td>
<td>683</td>
<td>71</td>
<td>400</td>
<td>0</td>
<td>tim</td>
<td>0</td>
<td>0%</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>SETTINGS</td>
<td>823</td>
<td>127</td>
<td>150</td>
<td>F</td>
<td>sus</td>
<td>0</td>
<td>0%</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>BUS.LCN</td>
<td>660</td>
<td>111</td>
<td>150</td>
<td>tim</td>
<td>0</td>
<td>3%</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>BUS.SAT</td>
<td>572</td>
<td>71</td>
<td>150</td>
<td>tim</td>
<td>0</td>
<td>2%</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>HISTORY</td>
<td>299</td>
<td>37</td>
<td>200</td>
<td>tim</td>
<td>0</td>
<td>0%</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SRV.REM</td>
<td>1029</td>
<td>200</td>
<td>300</td>
<td>sus</td>
<td>0</td>
<td>6%</td>
<td>6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>SRV.INT</td>
<td>603</td>
<td>138</td>
<td>200</td>
<td>run</td>
<td>0</td>
<td>2%</td>
<td>5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>LCN.SCN</td>
<td>943</td>
<td>188</td>
<td>400</td>
<td>tim</td>
<td>0</td>
<td>6%</td>
<td>5</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>LCN.RSP</td>
<td>707</td>
<td>123</td>
<td>150</td>
<td>sus</td>
<td>0</td>
<td>0%</td>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>SRV.CMD</td>
<td>1157</td>
<td>226</td>
<td>300</td>
<td>sus</td>
<td>0</td>
<td>2%</td>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>DELAY</td>
<td>283</td>
<td>33</td>
<td>100</td>
<td>tim</td>
<td>0</td>
<td>0%</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LOGIC</td>
<td>530</td>
<td>85</td>
<td>300</td>
<td>tim</td>
<td>0</td>
<td>0%</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SCHED</td>
<td>884</td>
<td>172</td>
<td>300</td>
<td>tim</td>
<td>0</td>
<td>7%</td>
<td>5</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>EVENT</td>
<td>985</td>
<td>210</td>
<td>300</td>
<td>sus</td>
<td>0</td>
<td>11%</td>
<td>4</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>SRV.PCK</td>
<td>665</td>
<td>130</td>
<td>150</td>
<td>sus</td>
<td>0</td>
<td>2%</td>
<td>4</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>SRV.CAM</td>
<td>521</td>
<td>87</td>
<td>150</td>
<td>run</td>
<td>0</td>
<td>0%</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>SRV.WEB</td>
<td>1865</td>
<td>366</td>
<td>500</td>
<td>sus</td>
<td>0</td>
<td>0%</td>
<td>5</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>SRV.DEA</td>
<td>907</td>
<td>164</td>
<td>250</td>
<td>run</td>
<td>0</td>
<td>3%</td>
<td>9</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>SRV.MOD</td>
<td>583</td>
<td>142</td>
<td>250</td>
<td>tim</td>
<td>0</td>
<td>7%</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SRV.UAV</td>
<td>1315</td>
<td>277</td>
<td>300</td>
<td>run</td>
<td>0</td>
<td>2%</td>
<td>9</td>
<td>26</td>
<td></td>
</tr>
</tbody>
</table>

**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