Process Data Connection Channels in uLan Network for Home Automation and Other Distributed Applications

Pavel Píša[1,2]
pisa@cmp.felk.cvut.cz
Petr Smolík[1,3]
petr@smoliku.cz
František Vacek[1]
fanda.vacek@volny.cz
Martin Boháček[1]
bohacma8@fel.cvut.cz
Jan Štefan[1]
honza.stefan@gmail.com
Pavel Němeček[1]
pavel.nemecek1@gmail.com

[1] Czech Technical University in Prague, Department of Control Engineering
Karlovo náměstí 13, 121 35 Praha 2, Czech Republic

[2] PiKRON s.r.o.
Kaňkovského 1235, 182 00 Praha 8, Czech Republic

[3] AGROSOFT Tábor s.r.o.
Harantova 2213, 390 02 Tábor, Czech Republic

Abstract

The uLan protocol is the multi-master communication protocol aimed on small RS-485 control networks. It provides deterministic media access arbitration and it is open in design from its origin. An open-source implementation of the protocol has already been available for many years. The article focuses on its adaptation for use in distributed home appliances (switches, lights and HVAC components interconnection and control). For resource restricted control nodes, it was a challenging task to implement a flexible and persistent configuration of data and events direct routing between distributed nodes without need for permanent operation of commanding master. Because devices do not have resources to mutually examine their often large objects/properties dictionaries, the mechanism to map properties values into process data messages slots has been implemented. The message slots act as (virtual) wires which are setup by configuration tools running on PC which has enough resources to build and visualize full objects/properties model by examining of connected devices. Examples of developed devices using developed concept are presented at the end of the article together with tools available to help with fast prototyping of new devices and their testing in PC environment. The compilation of embedded devices code as native Linux binaries is quite straightforward because uLAN driver implementation is portable and provides same API when compiled for system-less nodes, GNU/Linux or Windows operating system environment.

1 Introduction

There is a need for a cheap, two wires bus communication between resource constrained MCU based node in many projects and application areas. Many standards exist but most of them require a special MAC hardware to be integrated onto MCU or attached as an additional communication chip. Many technologies have the disadvantage of being proprietary or at least controlled by (sometimes secretly held) patents, even those declared as public standards. The described solution is based on a different approach. It is targetted to standard UART hardware (with multi-drop or stick parity bit support) available on most MCUs and PC serial port interfaces and it has been developed as an open protocol from its beginning.

The article is divided to main parts. The first one describes protocol basic ideas leading to uLAN protocol design and implementation. Description starts from low level frame protocol and describes uLAN Object Interface (uLOI) higher level layer with a brief application example.

The second part focuses on process data exchange based on device properties/variables values mapping into communication channels distributed in publisher-subscriber manner.

2 uLAN Protocol

Origin and Initial Target Applications

The protocol design has been motivated by the need of control and data acquisition networking suitable for next generation of High Pressure Liquid Chromatography (HPLC) instruments sets designed by yet future PiKRON company forming group in 1992. The HPLC chromatography instruments do not require so fast command/data exchange for basic setups, but there are many parameters which have to be setup and should be monitored. The data types range from simple one scalar variable setup (wavelength, flow rate) to gradient time program and detector data stream (one float at 25 Hz in our case). There should be no loss of sampled data but grouping into longer packets is possible (group of 32 samples at the time is used in our case). The requirement has the ability to send some synchronization commands between instruments without latency added by data resending or even polling cycle controlled by a single master (PC). Because the development of new/different instruments and components to the modular system was expected, the protocol higher level layers need to support examination of instrument type and available properties/variables.

uLAN Protocol Overview

The initial design of instruments control electronics has been restricted to Intel-8051 based controllers due to its availability and price. These devices provide only single UART hardware for communication and their computational power is quite low. But they offer multi-drop (9-bit per character) feature which allows to suppress the need to process these received data characters (address bit clear / bit 8 = 0) which are not a part of message targeted to given instrument/network node (module in uLAN terminology). uLAN defines character values $0 \cdots 0x64$ with address bit set to address target module but only up to 64 masters are considered by media arbitration described later. The value 0 works as the broadcast address. Values from $0x75 \cdots 0x7F$ range have control and data delimiters role. Values above 0x80 are used to release the bus by master after it finishes its mastering role in one message(s) exchange session. The whole range above 0x80 is used for bus release to encode releasing node/module address which allows to enhance fairness of communication channel capacity distribution between nodes. Due to standard UART behavior and need to synchronize on character basis, whole time to transfer character includes start and stop bit in addition to the address indication bit. The whole character transfer takes 11 bit times in uLAN case.

As a physical layer, RS-485 signal levels, wiring and transceivers have been selected. Because multi-master operation has been required (as stated above) some mechanism of media access control/arbitration has to be defined. The one solution is to use token passing (Profibus, BACnet MS/TP). But it requires to keep and update nodes lists in each communication node, initial single token selection and its regeneration after node failure is quite complex. RS-486 signalling does allow reliable collision detection on the wire when transceiver is switched to Tx direction. Switching between Tx and Rx direction and link level stabilization is much slower than available data rates as well. However simulation of dominant/recessive levels is possible by switching between Tx logic zero level and Rx direction when bus termination with quiet level bias to logic one is used.

uLAN deterministic distributed media arbitration has been partially inspired by Philip's I2C design. But to allow full speed data rates during message data phase and because UART hardware allows only control of transceiver Tx/Rx direction (in most cases assisted by CPU code in ISR) only on whole character time granularity, the arbitration is based on switching between Tx zero and Rx for whole character time (sometimes implemented by break character send). Not like in I2C case, the arbitration needs to finish before target address and data are sent in transceiver fully driven Tx mode. The arbitration sequence is based on self node/module address to ensure unique dominant/recessive sequence for each node.

uLAN is targetted to control applications which require data receiption acknowledgement and communication exchanges can be simplified by a direct reply by addressed device during a single arbitration cycle. Direct reply frame follows directly after initial frame end without media arbitration. Master releases the bus after last frame belonging to the given session. This is technique used in many other standards but the advantage of uLan is mechanism generic enough that there is no need to use specialized command format knowledge on the master's side of communication and required/expected single message session frames sequence can be prepared and passed to the driver on application level.

The single frame consists of destination address (DAdr) with address bit set, source address (SAdr), command (Com) followed by frame data characters. The end of data is delimited by one of four control characters describing the frame end kind. The simple frame consistency check byte (XorSum) follows. The frame is directly acknowledged if frame end kind specifies that. Then an direct reply frame can follow if indicated by frame end as well.

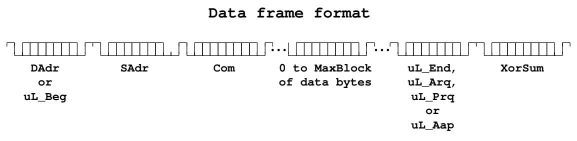


FIGURE 1: *uLan Frame Format*

Media Arbitration and Its Cost

The media arbitration is divided into two phases. The first phase is bus quiet time which given node waits to ensure that bus is free (T_{arbW}). The dominant level (break character) is sent after detection of T_{arbW} bus quiet time. If the other node character is received, arbitration restarts from the beginning. The second phase ensures a deterministic resolution for the case when two or more nodes finish the first phase in same time.

Because characters are sent and processed asynchronously (UART is used) and some delays could be caused by latencies in interrupt processing and some delays are even required for safe transceiver Rx/Tx switching without spikes the minimal time is specified as 4 character/byte transfer times T_{chr}.

The first phase T_{arbW} waiting time is not the same for all nodes to ensure some distribution of the channel capacity between multiple nodes. The wait time value is counted as

$$T_{arbW} = ((LAdr - Adr - 1) \bmod 16 + 4) \cdot T_{chr} \quad (1)$$

where $LAdr$ is node address of the last node which has won arbitration and now releases the bus, Adr is the address of given node which prepares for bus use and T_{chr} is time to transfer one character. This setup ensures strict cycling of media access priority between nodes with messages prepared in Tx queue when only addresses up to 16 are assigned to nodes. If more nodes are used, the cycling between aliasing nodes is not ensured on deterministic basis but at least helps with some stochastic distribution.

The second phase ensures that node with lower own address wins arbitration when two or more nodes finish the first phase at the same time. The arbitration is based on sending next three dominant level break characters separated from initial one by precomputed time intervals $T_{arb,0}$, $T_{arb,1}$ and $T_{arb,2}$

$$T_{arb,i} = ((Adr \operatorname{shr}(2 \cdot i)) \bmod 4 + 1) \cdot T_{chr} \quad (2)$$

If the activity from other node is detected during inactive interval time, the node abandons arbitration and restarts from the first phase. Direct binary coding and sending of own address as sequence of dominant recessive character intervals have not been selected because precise timing would be a problem through ISR responses. The addition of one dominant start bit and recessive stop bit around each arbitration bit would result in even longer phase two sequence ($3 \cdot T_{chr} \cdot 8 = 24 \cdot T_{chr}$) length.

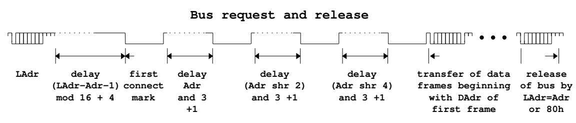

Bus request and release
LAdr
delay (LAdr-Adr-1) mod 16 + 4
first connect mark
delay Adr and 3 +1
delay (Adr shr 2) and 3 +1
delay (Adr shr 4) and 3 +1
transfer of data frames beginning with DAdr of first frame
release of bus by LAdr=Adr or 80h

FIGURE 2: *uLan Media Access Arbitration*

The designed deterministic distributed media arbitration poses quite significant cost and consumes important part of communication channel capacity. The 11 bit times T_b are required to transfer single character $T_{chr} = 11 \cdot T_b$.

The T_{arbAll} time of whole arbitration sequence ($T_{arbW} + 1 + T_{arb,0} + 1 + T_{arb,1} + 1 + T_{arb,2} + 1$) is bounded by next ranges

$$T_{arbAll} \in \langle 4 + 3 \cdot 2, 20 + 3 \cdot 5 \rangle \cdot 11 \cdot T_b \quad (3)$$
$$T_{arbAll} \in \langle 10, 20 + 35 \rangle \cdot 11 \cdot T_b \quad (4)$$

The whole time of one message arbitration cycle consisting of single frame and reception acknowledgement represents time interval $T_{arbAll} + (3 + l_d + 2 + 4 + 1) \cdot 11 \cdot T_b$ where l_d is number of data bytes. If network with only 10 nodes with addresses $1 \cdots 11$ is considered, the arbitration overhead is much lower due to shorter times of the second phase for modes assigned by lower addresses and because maximal length of the first phase applies only in case when same node requests bus repeatedly (see equation 1). The average message transfer time is more favorable for this case, if full Tx saturation from all nodes is supposed. The first phase time is $9 \times 5 \cdot 11T_b$ and $1 \times 13 \cdot 11T_b$ for this case. The second phase from 2 contributions evaluates to $7, 8, 9, 7, 8, 9, 10, 8, 9, 10, 11$ character times. The average arbitration time T_{arb} settles on $(9.6 + 5.8) \cdot 11 \cdot T_b$ and whole message time is $(l_d + 25.4) \cdot 11 \cdot T_b$. In case of quite common (for our HPLC applications) message length of 256 B and communication speed of 19200 Bd it takes 1.6122 s to send 10 messages (one from each station) and overhead caused by arbitration and other control characters represents 10 %. If the whole encoding schema is compared to synchronous communication which does not need any address, start and stop bits, the overhead causes 50 %. But even synchronous communication requires some bit-stuffing in real applications protocols and some media access control. On the other hand, if short messages of 8 bytes each are considered then uLAN protocol makes up much higher overhead about 300 % (550 % if counted on bit level).

The uLAN protocol compared to CAN can offer in case of dedicated or FPGA hardware solution up to 10 times higher transfer rates for bus of the same physical length because arbitration (requiring propagation of dominant/recessive level to whole link and back) is running with 11 times slower timing than actual data bytes. Other advantage is that during data transfer full active push/pull transceiver mode is used which provides better noise immunity and works well even if only single twisted pair of wires is used. CAN typically does not work well without ground interconnection. When compared to token passing networks, uLAN has much simpler (basically none) master node connection to the network and minimal delays are caused by node failure or switched off. The significant disadvantage of very high overhead for small messages can be adjusted by building higher level protocol in the way that multiple variables/properties transfers are grouped into single message.

Higher Level Layers

There are multiple higher level services built above low level uLAN messages and frames protocol described earlier.

Network Control Messages (uLNCS) the commands from this group allow to check and change module/node assigned network address, check its identification and production serial number

Dynamic Address Assignment (uLDY) the mechanism to unveil newly attached nodes from new serial product number appearance, assign them free network address and detect node disconnection or switching off

uLan Object Interface Layer (uLOI) the mechanism to retrieve list of device supported readable and writeable variables/properties, their names and data types

Only very short description of use of the last mechanism fits in this article.

3 uLan Object Interface Layer

The uLOI defines the system how to serialize objects (properties/variables) identification and their values in transferred messages. The service works with asynchronous reply as further master transfer after request sends service/command number to specific uLOI node/module. Multiple queries for objects values and/or their description can be serialized in a single message. The limitation is given only by the maximal length of a request and expected reply messages which is at least 1000 B for actual products. The controlling application can build model representing connected devices and then use this model to access data and control attached modules/instruments. The objects serialization and identification minimizes amount of metadata to minimize communication overhead. Each object in module is identified only by 16 bit Object Identification Number. No type, name nor data length for plain types is included in regular transfers. All these information has to be obtained by controlling application/system in advance through predefined OIDs for this purpose. The significant advantage of the protocol and current uLan driver implementation is that mesage can be read from incoming queue by parts and OIDs are directly interpreted and the reply message is build again in "driver space" buffers. The second advantage is that reply allows to identify which objects data it contains. This allows to have more data request on the fly from different controlling nodes or applications.

The example of system utilizing many of uLAN services is CHROMuLAN HPLC control system developed by Jindrich Jindrich and PiKRON Ltd.

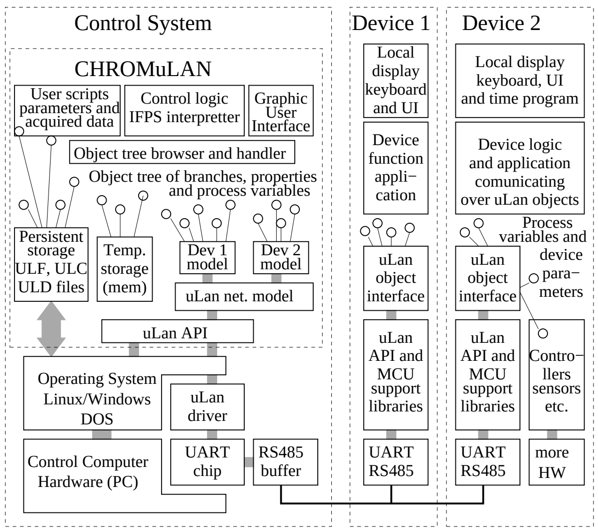

Control System
CHROMuLAN
User scripts parameters and acquired data
Control logic IFPS interpretter
Graphic User Interface
Object tree browser and handler
Object tree of branches, properties and process variables
Persistent storage ULF, ULC ULD files
Temp. storage (mem)
Dev 1 model
Dev 2 model
uLan net. model
uLan API
Operating System Linux/Windows DOS
uLan driver
Control Computer Hardware (PC)
UART chip
RS485 buffer
Device 1
Local display keyboard and UI
Device function appli-cation
uLan object interface
uLan API and MCU support libraries
UART RS485
Device 2
Local display keyboard, UI and time program
Device logic and application comunicating over uLan objects
Process variables and device para-meters
uLan object interface
uLan API and MCU support libraries
Contro-llers sensors etc.
UART RS485
more HW

FIGURE 3: *uLOI in Devices and Corresponding Model Build in CHROMuLAN Application*

Many other applications have been developed at PiKRON company or by other uLAN adopters. I.e. Agrosoft Tábor FASTOP and FASTOS systems for automatic batch distribution of feed to pigs, cows and their systems for cow milking data collection. uLAN interconnect the feeding units with RF animal identification with central database in these systems for example.

4 Data Exchange in Home Control Application

The multi-master capability of uLan, very low cost interconnection with use of a phone line grade cables, free bus topology, non problematic interfacing between many low cost microcontrollers and stable drivers for PC operating systems are features which speaks for spreading of uLan into other areas as well. uLan is not intended for high speed communication or hard real-time data exchange but these features are not required for the most tasks of home automation systems. That is why use of uLan for heating monitoring and control, lights switching and ringbells has been proposed by team preparing new home automation project at the Department of Control Engineering.

uLOI layer supports devices configuration and their state monitoring by higher level systems. But use of polling cycle by higher level system is significant disadvantage for home automation. The home appliances has to be equipped by system which allows direct communication between nodes in response to the incoming events. This is important not only to short latencies caused by polling cycle but even to allow system to provide at least basic functionality even in the case of higher level control system failure. It would be possible to use uLOI messages for direct data writes or reads to/from one appliance to objects located in other one. However, this would require mutual knowledge of the structure of appliances and require quite complex and memory resource huge OIDs list and types retrieval or made system inflexible by storing other device OIDs into firmware in fixed form.

The generic system for building uLan Connection Network (uLCN) for processing the data exchange has been designed instead. This mechanism consists of two main specifications. The first there is defined new uLAN protocol level command/service for process data exchange (UL_CMD_PDO). The message of this type contains one or more blocks holding data corresponding to individual virtual "wires" connected between appliances. Each such wire is identified by its Connection ID (CID) and delivers data or events of some type.

uLAN PDO Connection Channels

The subsystem is designed for direct process data (PDO) exchange between devices (nodes/instruments). Every data transfer is identified by connection ID (CID). Design allows to map one or multiple uLOI dictionary objects (properties, variables) as data source or destination for given CID. The mapping is stored directly in devices. The mechanism allows to transfer multiple CID identified data values in single message. Receiver identifies data scope only by CID, no source address or device internal uLOI OID assignment or meta-data format is encoded in PDO messages or directly influence the processing. This allows to connect objects with different OIDs, group multiple objects under

Res Lo	Res Hi	Ext len (el)	Ext	CID	data len (dl)	data	CID ...
1 byte	1 byte	1 byte	0..el bytes	2 bytes LE	1 (2) byte	dl bytes	

Table 1: UL_CMD_PDO Message Structure

a single CID, use broadcast to distribute data into multiple destination devices or even use more devices as data source for same CID. When device receives PDO message, it processes every CID identified data according to configured mapping. CIDs and their respective data for which no mapping is found are simply skipped. Only data types compatibility between mapped source and destination OIDs is required and sometimes this requirement can be even relaxed to some degree. If destination type is shorter then source, remaining bytes are skipped, counter case is illegal for actual implementation. Predefined constant data can be sent in response to event activation as well.

Command `UL_CMD_PDO` (0x50) is specified for PDO messages. Message format starts with two reserved bytes for future static extensions and one byte follows, which can be used for dynamic PDO messages header extensions in future. These bytes should be sent as zero for current protocol version. Each data block is preceded by its CID and data length. Maximal individual data block length is 127 bytes for actual implementation and is encoded in single byte. Format allows extension to two bytes in future if needed.

Control of Data Mapping into Channels

All configuration/mapping of PDO data source and processing of received PDO messages is done through device objects dictionary (uLOI). Exchanged data and meta-data stored in mapping tables have same format as is used for uLOI layer properties/data access.

The core component are ULOI_PICO and ULOI_POCO mapping tables, both with same format structure. They are accessible as regular uLOI arrays of four field structures. Each array entry specifies mapping between CID and object dictionary entries. Simple one to one mappings are specified directly by entry by OID number. Complex mapping can specify offset into block of meta-data byte array instead of direct OID specification. This allows to serialize multiple objects/OIDs data under one CID, add execute command after CID data reception and distribution into uLDOI objects etc. Another possibility is to process the same received data by multiple mappings for the same CID. The special form to embed 3 bytes (OID + single byte) or 4 bytes (OID + 2 bytes) directly into ULOI_PICO or ULOI_POCO mapping table entry is also supported.

Events to Process Messages Mapping

The ULOI_PEV2C array specifies, which CID/CIDs identified transfers should be initiated when given event number is activated. One event can be specified multiple times to trigger multiple CID transfers. The ULOI_PEV2C array entry specifies event number to CID mapping and some flags to nail down CID processing.

5 Example Applications

DAMIC Home Automation Components

The concept of the uLAN PDO connection channels is used in a components and appliances set which has been developed at the Department of Control Engineering to cover needs of heating, ventilation, air-conditioning (HVAC), light control and other home automation tasks:

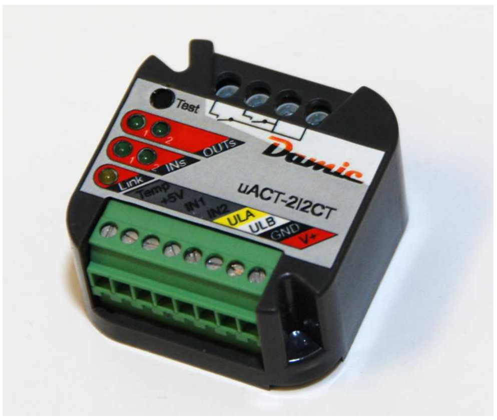


FIGURE 4: *uACT 2i2ct – uLan Actuator and Temperature Sensor*

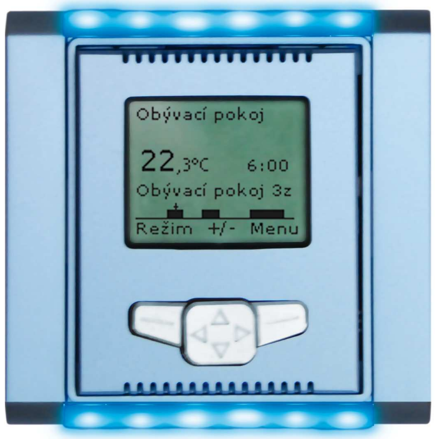


FIGURE 5: *uLTH 010 – uLan Room Thermostat*

uACT (010) an actuator and temperature sensor available in more variants of output and input channels count and power stages

uLMI (010) a device equipped by digital inputs to sense doors and windows state with additional temperature sensor

uLSW (010) not an only light wall switch which allows to map four contacts (left, right x up, down), their combinations and pres duration and “double click” to different events

uDIM 010 a multiple channels dimming controller for 8-230 VAC lights control

uLMO (010) a miniaturized variant of the actuator controller

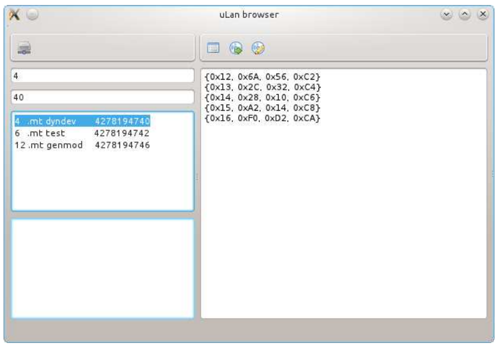
uLTH 010 a room temperature controller equipped by local multi-setpoint week program and user interface logic for program visualization and editing

All above listed components can be combined together. The temperature controller uLTH can control a heater equipped by valve controlled by uACT for example. The uLMI can be used to indicate open window and this state can be routed to the uLTH to switch of heating when inhibitant opens doors for ventilation. The designed infrastructure is used in a thermal recuperation and ventilation units (VECO) as well.

One or more computers can be used to monitor and visualize components states and setup parameters and time programs over uLOI protocol and or can participate in PDO uLCN based data exchange. The uLAN uLOI, uLCN infrastructure is used on PC hardware which runs Linux or Windows operating systems but even Linux equipped access-point devices or PowerPC based boards are supported by Linux builds of uLAN driver.

uLAN-Admin

uLAN-admin is a set of Qt library based components which provide access to devices properties/variables by means of uLOI protocol, allows devices scanning, identification, monitoring and administration. The core component is library ”libulproxy”. The model of uLan devices and their OI variables is built by the library in memory and creates abstraction to access uLAN network components over JSON RPC 2.0 interface. Thye library provides a routing of uLAN bus communication through TCP sockets as well. An utility library ”libulqttypes” take care about conversion of OI variables values between uLAN bus data format and Qt types. Its primary purpose is to decode/encode byte arrays of uLAN communication to/from QVariant variables. uLAN-admin also contains exemplary application ”browser” providing overview of devices on bus, which is based on above described libraries.



```
4
40
4  .mt dyndev    4278194740
6  .mt test      4278194742
12 .mt genmod    4278194746
```

```
{0x12, 0x6A, 0x56, 0xC2}
{0x13, 0x2C, 0x32, 0xC4}
{0x14, 0x28, 0x10, 0xC6}
{0x15, 0xA2, 0x14, 0xC8}
{0x16, 0xF0, 0xD2, 0xCA}
```

FIGURE 6: *uLAN-admin – ”Browser” Application Main Window*

uLAN-genmod

uLAN GenMod is an application that allows to connect a virtual devices to uLAN bus. Each device is defined by two files. A graphical representation of a virtual device is described by QML (Qt Modeling Language). uLAN description is defined in XDS file (XML description), where are device’s name, address, serial number and device’s object interface. A whole house network and variables interconnection can be configured by uLAN-admin tool through ULOI_PICO and ULOI_POCO tables, where is defined what PDO messages and CIDs device receive and send. The application allows save this network configuration. The network configuration is transferable to real devices. The virtual device can control the real devices connected to uLAN bus and vice versa.

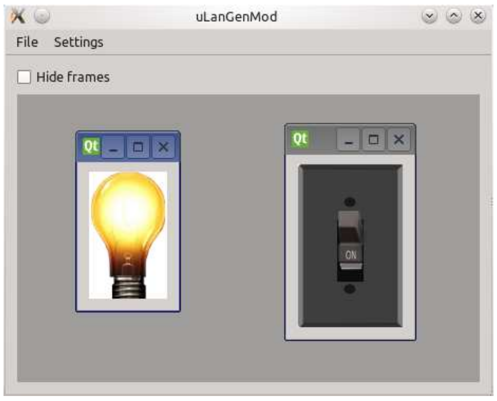


FIGURE 7: *uLAN-genmod – Application Main Window with Two Devices*

6 Conclusion

The uLAN protocol and surrounding infrastructure have been used in many applications for years. They include two generations of HPCL instruments sets (third generation is in preparation now), more agricultural control systems and componets, other serious production grade and hobbyists projects (i.e. HISC private house control network based on sole uLOI which componets has been designed around the year 2005).

uLAN uLCN/PDO design started in 2008 and its actual version is complete and well tested. The approach is similar to CANopen dictionary and PDO idea but it is more flexible and suitable for wider size data types, generic arrays and inherits underlaying uLOI layer flexibility. uLOI layer provides network introspection capabilities much better than many other standards offers. Yet the metadata overhead is kept very small for data exchange after initial device model retrieval phase.

The PDO mapping system has been tested on the CTU developed components for home automation during the DAMIC project. The initial versions of open-sourced management software utilizing Qt library is being developed as well. uLan driver and fully portable interface libraries allows to test even GNU/Linux builds of components and their interaction. The Qt based components builder and dictionary sources generator is in development to help newcomers to test capabilities and speed up new nodes design.

The uLCN/PDO mapping extension and openness of the project make it an excellent candidate for smaller hobbyists home automation projects. The minimal requirements for small nodes (only UART with software parity control) allows to base such designs on a cheap Cortex-M3 or even smaller MCUs. The design of higher communication layers can be utilized even in combination with different link technologies or can serve as an inspiration for other similar projects at least.

uLan project is a live thanks to more companies' and university members' participation. The actual version of the code used in multiple real sold products is available from uLAN project SourceForge GIT repository and file releases archives.

References

[1] Jindřich, J., Píša, P.: *CHROMuLAN project* [online], 2004–2011, Available: `http://sourceforge.net/projects/chromulan/`.

[2] Píša, P., Smolík, P.: *uLan Communication Protocol for Laboratory Instruments, Home Automation and Field Applications*, In *15th International Conference on Process Control 05*, Bratislava, 2005. Slovak University of Technology. ISBN ISBN 80-227-2235-9.

[3] Píša, P., Smolík, P.: *uLan SF.net project* [online], 2004–2011, Available: `http://ulan.sourceforge.net/`.

[4] Píša, P.: *ulan Driver and Protocol Base Documentation* [online], 2004–2011, Available: `http://ulan.sourceforge.net/index.php?page=3`.

[5] PiKRON s.r.o.: *HPLC Systems Manuals and Products* [online], 2011, `http://www.pikron.com/pages/-products/hplc.html`.

[6] Píša, P.: *Mathematics and electrical processing of liquid chromatography detector signal*, Ph.D. Thesis, Czech Technical University in Prague, 2010.

[7] Němeček, P., Čarek, L., Fiala, O., Burget, P.: *DAMIC – HVAC control system* [application prototype], 2009

[8] MIKROKLIMA s.r.o.: *DAMIC products for MIDAM Control System* [online], 2011, `http://www.midam.cz/categories/-DAMIC-inteligentni-dum.html`.

[9] *DCE MCU HW and SW Development Resources Wiki – Rtime Server at DCE FEE CTU* [online], 2011 `http://rtime.felk.cvut.cz/hw/`.