System Architecture for Distributed Intelligence

The usage of intelligence and sensors is not totally new, but solid-state lighting has brought it to a new level that has led to a new way of thinking about system architecture. The EnLight project found the distributed intelligence approach to be the preferable solution. Lex James from Philips Research, Martin Creusen from Philips Lighting, and Micha Stalpers from AME explain the reasons why, what the system architecture for distributed intelligence looks like, and how it works.

System architecture is about partitioning a system into its components and defining the relationships between those components. In general, it is good practice to partition a system so that the components are loosely coupled and have few dependencies while still maintaining the cohesion of the system as a whole. The notion of "separation-of-concerns" should be leading in any system architecture.

This is certainly true for the EnLight project which introduces a paradigm shift in lighting control by applying the "publish-subscribe" design pattern, appreciated both for its scalability and loose coupling. In such a lighting system, sensors inform luminaires instead of controlling them. As a result, the publisher (e.g. sensor) does not need any knowledge of its subscribers (e.g. luminaires). It just delivers its event to any subscribed device. This makes the publisher subscriber-agnostic, but what if the subscriber could also be publisher-agnostic too?

To achieve this objective, a second architectural choice is needed: every subscriber needs to be equipped with a decision engine that can be programmed with rules. The rules specify how each subscriber acts when it receives an event. So the rules have a dependency with the publisher rather than the decision engine itself.

The third important architecture choice for this project is closely related to the economy of scale of luminaire manufacturing. The concept of an Intra Luminaire Bus (ILB) allows luminaire manufacturers to build a large variety of luminaires from standardized HW/SW modules.

Since it would be impossible to address the complete EnLight system architecture in this article, the following three key topics have been selected. The chapter on system hierarchy addresses the various control levels of the system. The area level communication section describes the communication protocol that implements the "publish-subscribe" pattern. The third chapter covers the intelligent luminaire which acts as both a subscriber (light sources) and a publisher (embedded sensors). Due to the decentralized nature of the architecture, the intelligent luminaire is the cornerstone of the new developed lighting system.

System Hierarchy

Four distinct aggregation levels make up the EnLight architecture, as shown in figure 1. The project focused on luminaire and area levels, together with the corresponding intra- and inter-luminaire communication technologies. Building and enterprise, the two higher aggregation levels, are also supported by the new architecture (for example, to release information from the lighting system to the building management system).

The decentralized architecture is based on distribution of information rather than controlling individual luminaires via a centralized control unit. Thus "events" generated by sensors are distributed across the whole network. Thereafter each luminaire decides autonomously, according to its own embedded rules, how to control its lighting function based on notified events. So, decision making is made at the lowest aggregation level instead of in the centralized control unit.

Another key architecture element is the decoupling of the lighting domain from adjacent domains. Sensors report events without knowing the type of subscribed luminaires. Therefore, events are not interpreted or prioritized by the distributing sensor. This yields a transparent architecture with a

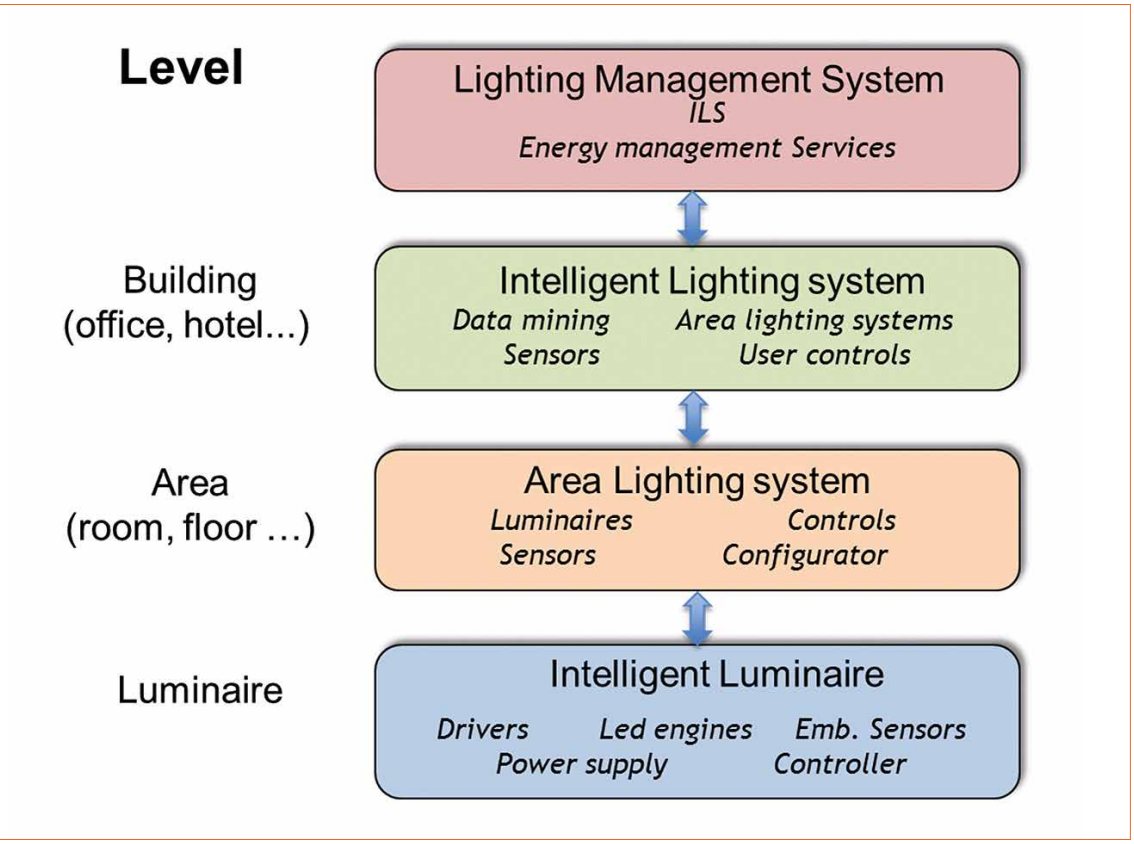

Figure 1:
Levels of lighting control

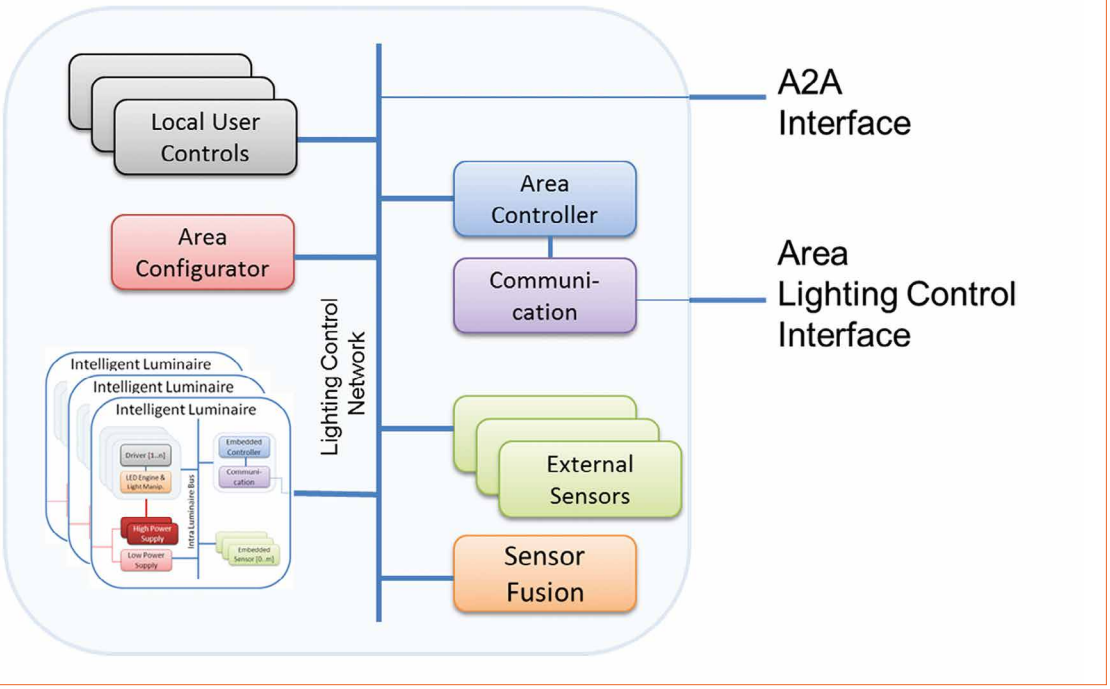

Figure 2:
Area lighting system

clear division of responsibilities in which the behavior of the main building block, the intelligent luminaire, is determined by the rules configuration - which can be easily adapted to a changing building environment. Additionally, self-learning algorithms would typically be accommodated by the intelligent luminaire.

In principle, any external device that can generate an event is capable of informing luminaires and therefore interacting with the lighting system. The event source does not require any upfront knowledge of the luminaires, enabling a smooth integration with building management systems.

On the Area level, the lighting control network connects the intelligent luminaires and key components, such as local user controls and external sensors (Figure 2).

The Area configurator enables the configuration of these components simply by changing the rules-based

logic of the intelligent luminaires. Optionally, events generated by multiple external area level sensors can be aggregated and processed using the Sensor Fusion function to reduce the network's overall communication load.

For monitoring or logging functions, the area controller and communication unit can send information from the area network to the higher building level and vice versa. Furthermore, the area controller can transfer events generated at different network levels. Finally, the area controller can also propose a certain light control function, such as an orchestrated color change, at the area level.

In the final EnLight demonstrations, the lighting control network was based on wireless ZigBee technology, as this ensures a scalable low power and low cost implementation. Wired implementations are also possible, but were not implemented in the demonstrations.

The main objective of the project was to develop a lighting system that offers excellent user comfort at the lowest possible energy consumption. This requires the right light to be generated in the right place at the right time. This was achieved by adopting granular sensing and control in a connected lighting system.

Granular sensing ensures, for example, uninterrupted luminaire operation during network interruptions. Granular control comes from embedding most of the decision logic in the luminaires, instead of using a centralized controller to send commands to individual luminaires. The luminaire is intelligent because it autonomously controls its own light sources. Thus the lighting system is effectively a collection of connected and cooperating intelligent luminaires which are able to adapt to changes in the environment.

Area Level Communication

Communication between devices within an area plays a prominent role in the EnLight architecture. It is designed to run on top of existing communication standards like ZigBee and/or IP and to support scalability and flexibility of the lighting system. The communication function is implemented as a set of reusable modules to guarantee consistency and facilitate ease of integration.

Communication protocol

The communication protocol is designed in such a way that new sensors and luminaires can be added to the area network while other devices remain operational. System intelligence, in the form of rules, can be configured and simulated off-line. Offering intelligence by configuration is a key element of decentralized systems and accommodates the system with a number of unique properties.

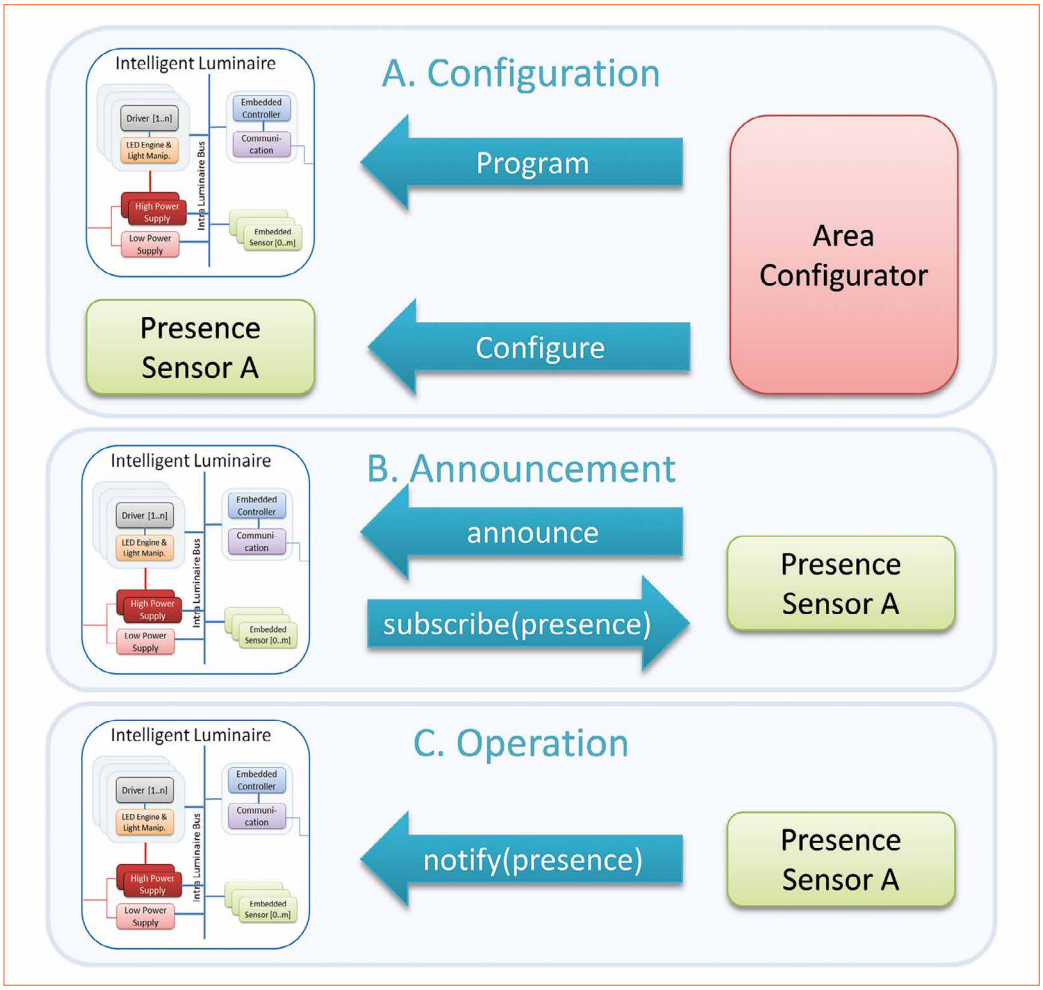

Figure 3:
Example of including a new sensor in the area network

- In a building environment, new sensors can be installed and put into operation without needing to take the whole system offline. Luminaires requiring sensor input for their decisions can be easily updated with new rules in just a few seconds.
- Once installed, new luminaires can be quickly put into operation by connecting the power and providing them with their initial rules.
- All EnLight devices are interconnected - there is no dependency on a physical topology. This allows unlimited intelligence reconfiguration simply by providing the luminaire with new rules. Reconfiguring an area with 40 luminaires only takes a few minutes.
- Reconfiguration of a single sensor, luminaire or part of the system is possible without impacting the rest of the system.
- As the impact on uptime is very limited, automation of the reconfiguration process is possible allowing for self-learning capabilities.

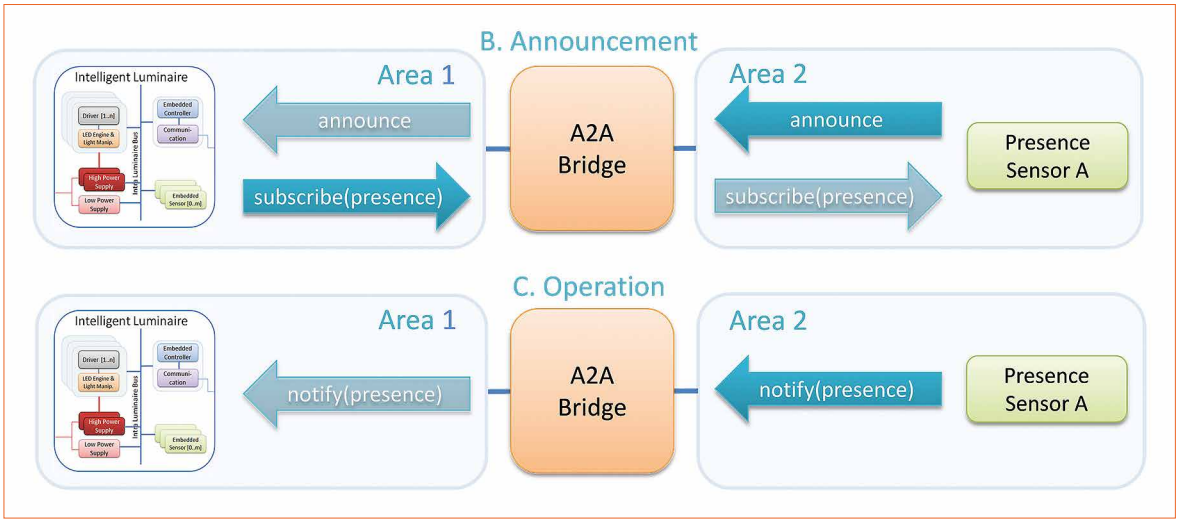

Figure 4:
Example of a bridge interconnecting two area networks

The communication protocol is best explained by the example outlined in figure 3 where sensor A is newly added to the system. First the area configurator configures the sensor by sending one or more parameters (e.g. absence delay). Since the sensor is new, the area configurator will also program one or more rules into the luminaires telling them how to respond to events from the sensor.

After configuration, the sensor announces itself to the network. Announcing is typically initiated by a button press or internal timer. When receiving the announcement, each intelligent luminaire checks the occurrence of the sensor in its rules. If it is found, the luminaire sends a subscription request. As a result the sensor adds the luminaire to its subscription list.

After subscription, the newly added sensor is now operational within the area network. As soon as the sensor detects something of interest, it sends an event message to all devices in its subscription list. Each luminaire receiving an event that matches one (or more) of its rules will execute the associated actions. More details are given in the next section.

The protocol separates the sensor domain from the luminaire domain. Devices other than luminaires can send events without adding lighting-specific details. They can also subscribe to events for monitoring and analysis. Information provided by the system can be valuable in determining energy saving strategies. Examples are “heat maps” of occupancy information and daylight measurements.

Bridging areas

The Area-to-Area (A2A) bridge has been developed to interconnect networks. The bridge can connect networks based on different communication technologies (e.g. ZigBee to IP bridge) but can also be used to scale a network to any size independent of communication-technology constraints. The bridge acts as proxy for devices on other networks and is therefore invisible for the light designer.

Figure 4 outlines how the bridge interconnects two area networks for a presence sensor and an intelligent luminaire programmed to respond to the sensor’s events.

As described previously, including devices in the network and device configuration is performed locally at area network level. The protocol was designed to allow inter-area communication for announcement and operation by making a distinction between a message’s local source and its origin in the system.

Communication framework

The communication framework was developed with three objectives in mind: ensure a consistent protocol implementation across devices from multiple partners, ease integration and minimize development effort. The framework provides the “glue” between an EnLight C-API and the EnLight binary protocol. It enforces a clear separation between application development and communication technology.

When an instruction from the C-API is invoked, the framework encodes a message containing the corresponding command and sends this to a given target device in the network, identified by a unique 8-byte device ID. It assumes the existence of basic communication operations “send” and “receive” for a particular technology. It also converts device IDs to technology-specific network addresses, such as ZigBee short IDs or internet IP addresses. The response received from the target device is decoded by the framework and results in the corresponding API callback to the application.

The main benefit of this framework approach is that the complex communication layer is made available to application builders via a simple interface without needing to know all the details underneath.

The ZigBee technology layer was implemented for a Windows environment using a ZigBee dongle, with a number of sensors running on an EM250 (Silicon Labs) and the luminaire running on a JN5168 (NXP). For security, the Common Security

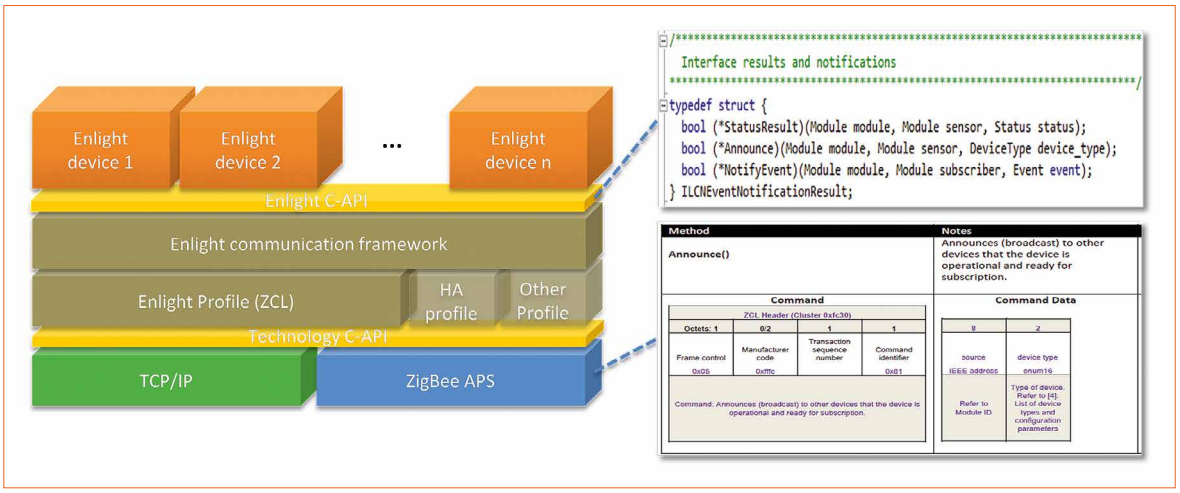

Figure 5:
Communication framework layering

Model with a predefined link key was used. Conforming to the ZigBee cluster library format, EnLight's ZigBee implementation can coexist with other ZigBee profiles such as home automation.

Interface details

The communication framework offers a set of interfaces that can be regarded as clusters of commands which will be described individually.

The device configuration interface offers run-time configuration of sensor and luminaire parameters in an operational system. To avoid inconsistencies during configuration, the configuration start and end are explicitly communicated. Getting and setting device parameters is based on key-value pairs. Moreover, the list of parameters supported by a particular device can be requested, and only the Area Configurator requires knowledge of these parameters. This allows a particular device's parameters to be extended without any impact on the protocol structure.

The programming interface offers run-time configuration of the luminaire intelligence. The rules are encoded into a compact binary format and transmitted to the luminaires in packets.

The event subscription interface is typically implemented by (but not limited to) a sensor device to allow other devices to subscribe to its events. It also allows other networked devices to request a sensor device to announce itself, enforcing announcement after system reconfiguration.
The intelligent luminaire uses the interface to subscribe to sensor events. The notification command is used to inform other devices of an event.

Table 1:
Device configuration interface

Method	Notes
StartConfiguration()	Starts configuration of the device
GetParameterList()	Returns the list of parameter identifiers the device supports
GetParameter()	Returns the value of a parameter, gives its identifier
SetParameter()	Sets the value of the parameter with the given identifier
EndConfiguration()	Signals that configuration of the device is complete

Table 2:
Programming rules interface

Method	Notes
StartProgramming()	Starts programming of rules for the device
AddRuleFragment()	Adds (a fragment of) a new rule for the device
EndProgramming()	Ends programming of rules for the device

Table 3:
Event subscription & notification interface

Method	Notes
SubscribeToEvent()	Subscribes to the given event of this device
UnsubscribeFromEvent()	Unsubscribes from the given event of this device
RequestToAnnounce()	Request a device to announce itself
Announce()	Announces that the device is operational and ready for subscription
NotifyEvent()	Notifies the occurrence of an event to the subscribers

Intelligent Luminaire

Luminaire architecture

There are diverse contemporary luminaire designs across various, regionally different lighting market segments. A main architectural question is therefore how to manage such a scattered mass-volume market cost-effectively. To avoid specific (low volume) peak designs, with relatively high development and bill-of-material (BOM) cost, common modules and standardized interfaces need to be (re-)defined as the lighting industry has always

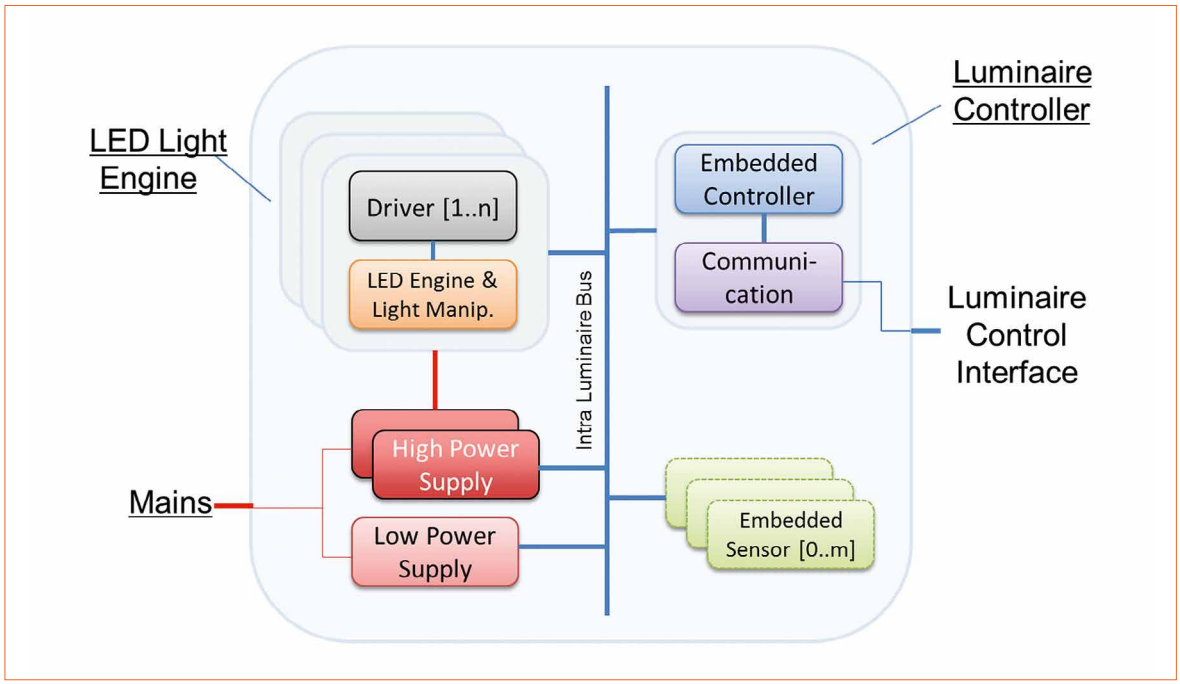

Figure 6: Modular luminaire block diagram

been based on standardized light sources and interfaces (e.g. E27, E14, GU10, etc.).

To date, standardized interfaces for LED-based light sources have been scarce. However, the global lighting consortium, Zhaga, published its first interchangeability standards in 2012. Zhaga standards focus on non-competitive aspects of the main interfaces, ensuring sufficient design freedom to differentiate and compete. This will lower the adoption hurdle for LED design-ins and further fuel the LED-based lighting industry.

The EnLight luminaire architecture uses an I^2C-bus interface to interconnect embedded components, allowing partners to contribute their technologies. The resulting modular, Plug-and-Play architecture facilitates future luminaire upgrades. It also enables 'late stage' luminaire configuration, either during production or installation. Finally, an easy, exchangeable modular concept reduces potential cost-of-non-quality, and decouples the lifecycles of independent technologies. The schematic diagram in figure 6 depicts the components of the architecture's key building block: the intelligent luminaire.

The different functions in the diagram can be either integrated or stand-alone, as preferred. For example, it is possible to combine the driver and LED engine, or the embedded controller and LED engine. The distributed power architecture uses centralized high- and low-power supply units, supplying 24 V DC (i.e. SELV) to the LED modules and drivers (optionally, the 24 V DC supply can also be used for power-demanding embedded sensors), and 5 V for the communication bus. It centralizes the power factor control and harmonic distortion circuits, and ensures low voltage Plug-and-Play operation for the luminaire's other key components. This reduces design complexity significantly, especially in multi LED Light Engine (LLE) luminaires.

Embedded sensors, such as temperature, light level or absence sensors, increase control effectiveness as they are typically co-located with the luminaire. This also simplifies installation and commissioning.

The embedded controller implements the decision engine and logical luminaire. The communication module connects the embedded controller to the 'Lighting Control Network' (LCN).

LED light engine

The LLE actually produces the light and is thus the module with the largest diversity. In a full color luminaire like the Wedge, the LED engine differs significantly from the single channel LLE used in the TaskFlex. The LLE's key functions are depicted in figure 7.

The color generation module calculates the individual primary colors intensity to realize the closest color possible. Because some applications require slow color transitions combined with a fast response to brightness changes, the LLE has a separate brightness and color fader.

Embedded controller

LLEs and Embedded Sensors need a controller to bring the luminaire to life. From a software perspective the controller is the most complex module. Fortunately, all EnLight luminaires share the same controller software, so development cost can be spread over many luminaires.

The "Decision Engine" is the embedded controller's brain and the attached memory contains a set of rules which determine the luminaire's behavior. Rules are programmed over the "Luminaire Control Interface" and stored in persistent memory.

Figure 7:
LED light engine block diagram

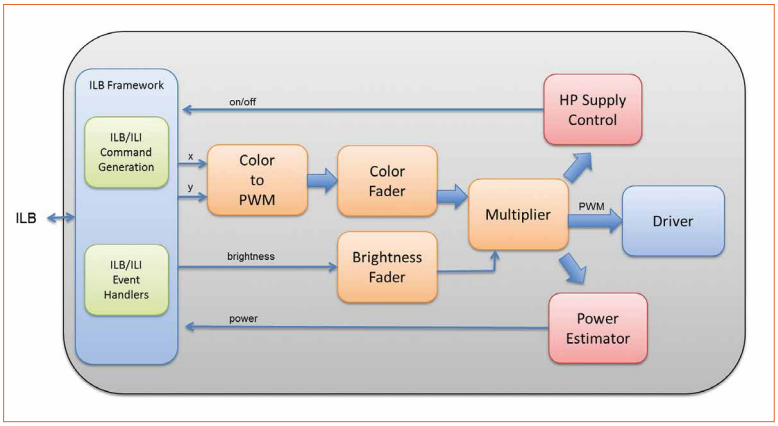

Figure 8:
Embedded controller block diagram

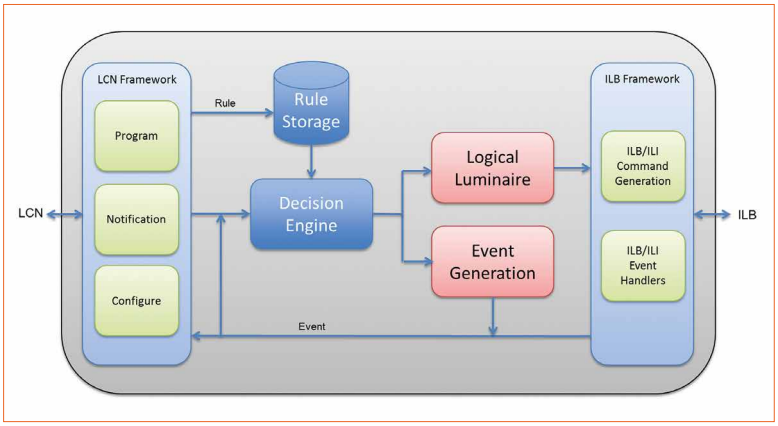

Table 4:
Rule actions (some actions require arguments, comprising the event's parameters and variables - Figure 9)

Action	Explanation
LevelActivation	Activates or deactivates a priority level
SetVariable	Set the content of a variable
LuminaireSetting	Change a light setting of a LED Light Engine
SetTime	Set the local time of the luminaire
GenerateEvent	Generate an event

Table 5:
Luminaire commands

Luminaire Setting	Explanation
On	Switch the light on
Off	Switch the light off
SetColor	Set the color
SetBrightness	Set the dimming level
ActivatePreset	Apply the settings according to the specified presets
SetLightLevelControl	(De-)activate the closed loop light level control
SetColorFadeTime	Set the time to go from one color to the other
SetFadeTime	Set the time to go from one dim level to the other
SetPowerMode	Set the high power supply mode: on, off or auto

The decision engine can be triggered by both internal (ILB) and external events (LCN). Each time an event is received all the decision engine's rules are evaluated, and if a rule is triggered one or more actions are executed. Typically, an action results in a luminaire command that is passed to the logical luminaire. This component translates a luminaire command into ILB messages to control the LLEs.

Decision engine & rules

In general there are two methods for implementing decision logic: scripting or rules. The consortium chose a rule-based solution as most lighting scenarios map well onto rules. Rules are also simple and inherently parallel, so lighting designers are not forced into the sequential thinking of programmers. The decision engine evaluates rules upon receiving an event. Rules are kept relatively simple to allow implementation on low cost hardware. While specified in XML, rules are programmed and stored in binary format and consist of three parts: Trigger Event, Condition and Action(s). The Trigger Event specifies the event (type and source address) that will trigger the rule. If an event matches the rule's trigger event, its condition is tested; if true the specified actions are executed. Table 4 lists the supported actions.

Logical luminaire

The logical luminaire understands "priority levels", with all settings applied on a certain priority level. New settings only become visible if the specified level is activated. If more priority levels are active simultaneously, the highest level settings will be used. The level concept offers a powerful means to prioritize conflicting settings. The "LevelActivation" action is used to activate or deactivate priority levels. As shown in the example (Figure 9), a level's activation can have a timeout and when the timer expires, the level returns to its original state. This feature is often used to make rules more robust for temporary communication flaws.

The most important luminaire commands are listed in table 5.

The "SetLightLevelControl" command activates the 'closed loop light level control', often used for daylight harvesting. The luminaire will try to realize the specified amount of light as measured by the light level sensor.

The logical luminaire also takes care of presets and "Level Definitions", which can be configured and stored

in persistent memory. Presets define each LLE's brightness, color and on/off state, and can be recalled with the "ActivatePreset" command. Similar to a preset, Level Definitions allow default luminaire settings to be overridden at power-on.

Event generation

A luminaire can generate its own virtual events and the generic implementation of this feature makes it very powerful. Events can be generated instantly or after a specified time. This effectively implements a one shot timer, commonly used in lighting applications. For scheduled events, which require an absolute time and date, the luminaire's real time clock ensures the event occurs at the correct time. Finally, the event module can be configured to generate events on a periodic basis.

Key specifications:

- Local rule based engine, up to 250 rules
- Up to 250 presets
- Color control: CCT, xy, HSV, RGB
- Up to 16 priority levels
- Embedded sensors: PIR / Light Level / Temperature / Switch
- Closed loop light level control

Conclusions

The consortium was a unique collaboration with partners and, sometimes, competitors.
The following conclusions are based on scientific and technical results. Commercial feasibility has not been addressed and is not taken into account in the conclusions.

```
<Rule>
  <TriggerEvent Type="SceneSelected" Address="00:04:74:00:00:6B:3C:F8"></TriggerEvent>
  <Action>
    <LevelActivation>
      <Level Activation="true" TimeOut="10">4</Level>
    </LevelActivation>
    <LuminaireSetting Command="ActivatePreset">
      <Level>4</Level>
      <Argument>
        <EventArgIndex>1</EventArgIndex>
      </Argument>
    </LuminaireSetting>
  </Action>
</Rule>
```

Figure 9:
XML presentation of a rule - The event "SceneSelected" is generated by a 6 button panel. The event comes with one parameter representing the pressed button number. This parameter is passed to the "LuminaireSetting" command "ActivatePreset". So the shown rule will activate a preset depending on which button has been pressed

The EnLight architecture scales smoothly from a single luminaire to many hundreds. This makes the luminaire best place to host the intelligence (decision logic) of a lighting system as the amount of intelligence in the system grows linearly with the amount of luminaires.

Using a standardized intra luminaire bus enables luminaire diversity based on a limited amount of standardized modules from different vendors. The consortium realized 14 types of luminaires, all based on the same building blocks. For these luminaires, a simple yet powerful decision engine (+ZigBee stack) was implemented on a low cost microprocessor (16 MHz RISC, 32 kB Ram, 256 kB flash). Further, the ILB framework showed that software reuse is possible when moving from modular hardware to fully integrated hardware (cost down for high volume).

Informing as opposed to controlling a luminaire implies that sensors and controls don't require any knowledge of the system's luminaires. Embedded sensors in combination with local intelligence allow simple and robust granular control, but a well thought-out communication framework is needed for the effective development of such a system. In this case, the use of a publish/subscribe design pattern results in very efficient communication and combined with the use of prioritized control (levels) delivers a very powerful solution for lighting systems with multiple sensors and controls. ■