INTERRUPT INSIDE

An in-depth magazine about IoT and embedded technology from Data Respons 1, 2017

EnergyBASE:
IoT solution for innovative energy management

AI / Machine learning:
Image classification using artificial neural networks

Wireless:
A shortcut to Smart Mesh Networks

data respons
EMBEDDED SOLUTIONS

WHAT'S **INSIDE**

04 **EnergyBase: IoT-based solution for innovative energy management**

EnBW, one of Germany's biggest producer of electrical energy is driving a project which helps the typical home owner to migrate from mere energy consumption to becoming a "prosumer" (producer-consumer) with local power production and storage by solar panels and stationary batteries.

10 **Fake?**

This article discusses the risks and challenges manufacturers meet when faced with counterfeit components.

14 **Agile system modeling**

The concept of modeling system requirements and design in is not a new one. However, recent advances in languages and tools has created opportunities for reducing the total development effort for embedded systems, and improve quality. This article aims to present some of these opportunities, based on the authors' experiences.

19 **A shortcut to Smart Mesh Networks**

In the emerging world of Internet-of-Things, wireless low-power mesh networks are more relevant than ever. Data Respons has gained valuable experience with one particular technology after employing it in a large industrial instrumentation project, namely Linear Technology's SmartMesh IP. As a specialist on embedded solutions, Data Respons recently became an official Linear SmartMesh partner, after developing the QuickStart Library: A software library that greatly reduce development time for embedded applications of SmartMesh IP.

22 **Image classification using artificial neural networks**

Unlike in traditional computing, neural networks can be designed and trained to identify complex patterns without specifying the pattern-features in an algorithmic manner with if's and else's. Image Classification is one of the fields where neural networks can be put to use.

EDITOR

(R) Evolution

Everything is getting digitalised, connected and automated. At first glance, this may look like a revolution because of the great impact in all industries. However, this is not new –we have continuously been digitalising the last three decades, connected in larger scale the last 50 years and automation has been around for centuries. The constant evolution of technology drives the development.

Data Respons has been working with the enablers of digitalisation, IoT and AI (machine learning), involving data collection, connectivity and computer processing through embedded technologies, sensor based data acquisitions systems and complex SW development for more than 30 years. Smarter utilisation of the data surrounding us is key driver for most of our customers – we are a complete technology partner in this field and our competence stretches from the sensor level to the final APP.

In this issue of Interrupt Inside we will look at how smart home management technology create economic opportunities for the consumer using IoT technology, smart device software and data collection to make informed choices on when it is optimal to sell or store your harvested energy. We investigate the use of machine learning to recognise and identify serious eye illness through traditional image classification methods. In addition, we are digging into the counterfeit component industry.

All articles presented in Interrupt Inside are written by our own specialist engineers and select guest writers. We welcome any feedback and suggestions from our readers.

Enjoy the reading!

KENNETH RAGNVALDSEN

ENERGYBASE

IOT-BASED SOLUTION FOR INNOVATIVE ENERGY MANAGEMENT

BY: Thimo Koenig
Senior Software Developer
MicroDoc GmbH

BY: Christine Mitterbauer
Senior Engineer and
Project lead "EnergyBASE"
MicroDoc GmbH

The market for electrical power has changed substantially during the last couple of years. Central power plants are getting less important while decentralised power production of regenerative energy becomes more common. The so called "Energiewende" in Germany, a political program to shut down nuclear power plants and foster solar power and wind farms, has fueled this development and allows private households to sell energy into the public power grid. EnBW, one of Germany's biggest producer of electrical energy is driving a project which helps the typical home owner to migrate from mere energy consumption to becoming a "prosumer" (producer-consumer) with local power production and storage by solar panels and stationary batteries.

"EnergyBASE" represents a paradigm shift for EnBW. Rather than just selling power to their customers, EnBW is now offering technology and process know-how for energy management.

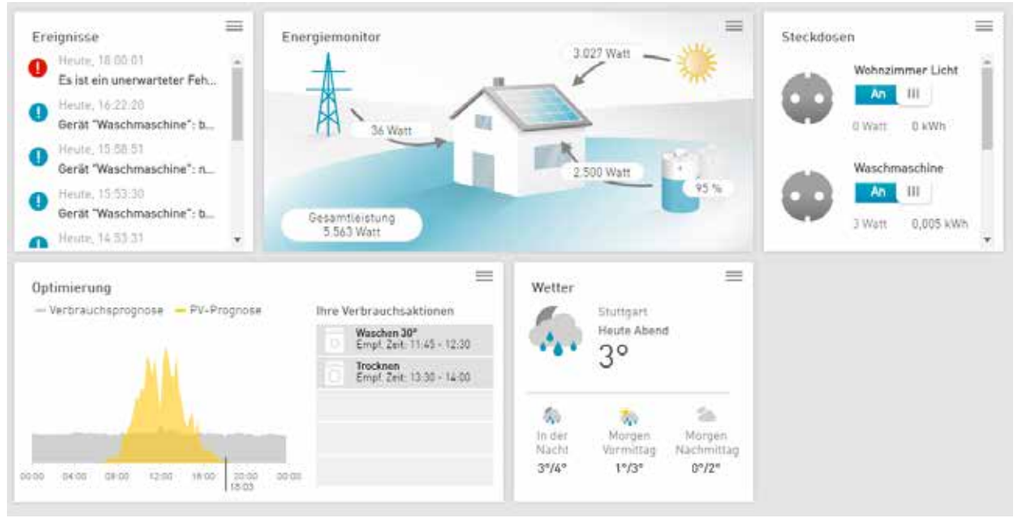


Illustration 1: EnergyBASE FrontEnd

ENERGYBASE SYSTEM COMPONENTS

The EnergyBASE system is comprised of the intelligent edge component (EnergyBASE hardware and software stack), an optional backend that integrates a variety of services, and a modern web UI provided by the local webserver on its EnergyBASE device. This customer frontend is optimized for desktop and mobile devices. As shown in the Illustration 2, the frontend is accessible through the local network and (if activated) also from remote.

ENERGYBASE DEVICE

The EnergyBASE device contains a 450 MHz ARM9 processor, 128 MB RAM and 4 GB of flash storage. It provides ethernet and serial interfaces and also a polyphase electric meter. The EnergyBASE software stack is made up by a MicroDoc port of Oracle Java™ SE Embedded 8 JRE and a Smart Home OSGi framework.

ENERGYBASE DEVICE SOFTWARE ARCHITECTURE

The EnergyBASE applications are written in Java. This allows for portable code for a variety of target platforms. While the project teams develop on Windows, Linux and Mac system, the same Java code can be used on cloud based test- and demo instances as well as on the actual target hardware of the EnergyBASE without any code changes. The

Illustration 2: System components

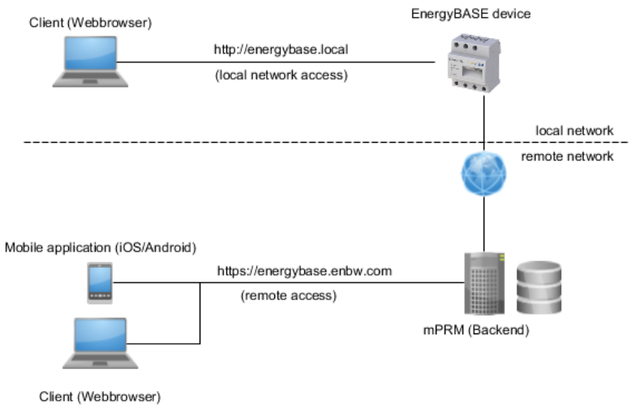
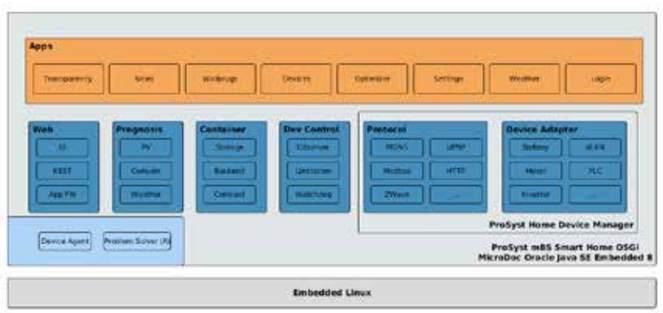


Illustration 3: EnergyBASE architecture



>>

Java promise "write once, run anywhere" is real. The EnergyBASE architecture is also based on the OSGi component model. By choosing this technology, we are able to provision, deploy, start, stop and remove software components (called "bundles") on-the-fly on remote devices without interrupting operation or other services on the device.

Bundles can be updated individually or within groups, which gives us the ability to react quickly and effectively to new requirements and potential problems in production environment.

The component model is used to assemble customer specific applications depending on parameters like hardware release, customer contracts, configuration, stage or use case scenario. As shown in the Illustration 3, there are many possibilities to combine the application bundles: multiple adapters to handle different kinds of devices from different manufacturers, implementation of protocols for communication purposes, external service connectors, selectable forecast algorithms, optimization methods to use the energy in an efficient way and much more.

There are a few preliminary decisions you'll want to make while defining an OSGi bundle. One of these decisions is the dependency to other bundles. Each bundle can be independently defined or in conjunction with other. For example, let us assume Bundle B is dependent on Bundle A. In this case, it is sure that the startup process of bundle B will be initialized after bundle A is already in the correct state (started). In more concrete manner: it will not happen that one of the device adapter bundles get started while the necessary protocol implementation bundle is not available.

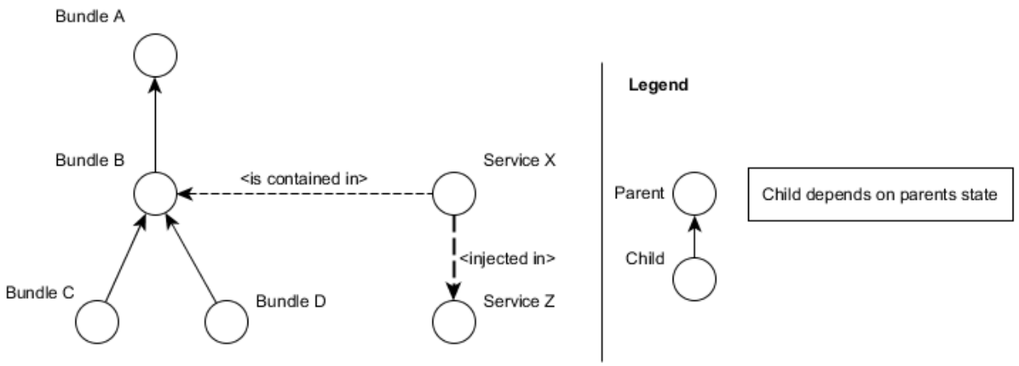
Furthermore, we developed a mechanism to define relationships between services and the ability of injection. According to the inversion-of-control pattern, our ServiceMonitor (or more specifically the OSGi BundleContext) observes and manages the complete lifecycle of each service and provides the requested instance. At this point the relationship between dependencies on Bundle- and Service-Layer becomes much more important. Following the Illustration 4 we can see, that Service X is injected in Service Z. The instance of Service X can only be created when Bundle B is running. Due to the relationship between Bundle B to Bundle A, its running state is also necessary. This small example shows that this technique give us a handy way to control dependencies but can also grow to a complex construct really fast. In practice we keep the dependencies as small as possible.

> This feature is very convenient to develop automated test-cases which implement complete test scenarios over all system components.

Despite having the loosely coupled components, the ability to communicate to each other by means of an event-based publish/subscribe mechanism is still present. In addition to the general properties of the OSGi-based platform, we also benefit from various add-ons created by the OSGi-Engine. It provides several features to monitor and manage external devices. The EnergyBASE behaves in a very performant and smooth way notwithstanding to the huge amount of functionality of the engine and the complexity (see Illustration 3) of our application.

The ability to communicate with our backend through a (SSL encrypted) TCP socket connection is already provided by the used OSGi engine. The EnergyBASE is obviously completely useable without an active connection to the internet or our backend. But there are some handy features, also used by most of our customers, like remote access through the web (https://energybase.enbw.com), mail sending in case of malfunctioning or weather consumption which requires an active connection.

Illustration 4: Bundle and Service dependencies



ENERGYBASE BACKEND

Our backend system is based on the "mPower Remote Management" (mPRM). It is built by using the same software stack Java/OSGi, as the EnergyBASE, which brings us many advantages. It provides some essential features out-of-the-box like monitoring external devices, configuration, remote software updates and the internal repository to handle different versions of software components. We are able to extend the existing set of functionalities by providing self-developed bundles. We use this technique, for example, to consume weather-data or electricity prices from external service providers, sending emails and push notifications or to activate and deactivate EnergyBASE(s).

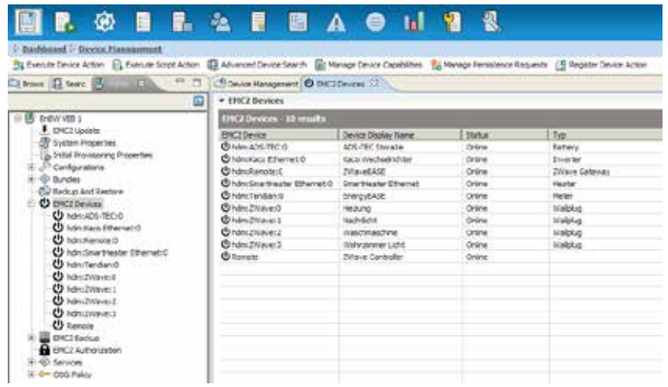


Illustration 5: EnergyBASE backend (mPRM)

The mPRM provides a generic RESTful API which allows executing its functions via HTTPS service calls. This feature is very convenient to develop automated test-cases which implement complete test scenarios over all system components.

Due to the use of Java and OSGi on server and device side, it's easy to implement distributed services for both components. For example, in case the EnergyBASE is connected to a server, weather data can be consumed and prepared on the server-side, while the device is only collecting the relevant regional data from the backend. Furthermore, it is possible to shift functions from the EnergyBASE device up to the server to process computationally intensive operations.

Further aspects of the general software development process are affected by the homogeneous choice of technology. We can use the same IDE, with the same set of plugins and also the same testing framework to develop client and server bundles. Also, the build and publishing process on our CI system does not need any changes. This may not sound very important but when you have already worked with totally different technology stacks on the client and server side you will be very pleased with the simplicity of this approach.

FRONTEND APPLICATIONS

The frontend applications are implemented using modern web-app technologies and are provided to allow end customers access to statistics and process control via desktop and mobile browser. Besides its browser-based access there exists also hybride, HTML5-based mobile web applications.The set of functionality is not as large as the default web application but it contains all important data to get a broad overview about the current energy production, batteries state of charge and further more (see Illustration 6).

Each UI related bundle contains three directories to provide its content for different environments: one folder just for mobile application related files, one for web browser files, and one shared folder for both cases. The CI system chooses the right files in conjunction to the target environment while building the software.

The mobile applications are currently available for Android and iOS as well.

Illustration 6:
EnergyBASE Mobile App

VIRTUALIZATION & TESTING

We are able to virtualize every component of the whole system, including the devices and the device adapters. This technology is very useful when it comes to testing individual device configurations as well as performing integration testing. Our continuous build process makes use of automated testing during nightly builds. It is also possible to model complete sample installations (virtual households) that can be used for training of the system maintenance staff or to support the sales process: it is always very convincing to demonstrate a live system rather than showing off a slide deck.

CREATING A "WHITE LABEL" SOLUTION FROM A BRANDED PRODUCT

Our customer EnBW decided to offer a "white label" variant of the EnergyBASE product due to market demand. The main challenge for an OEM product provider is to allow for flexible extensions, customization and customer specific skinning of the applications. Our aim was to provide mechanisms for customizing the EnergyBASE software and offer customers limited or changed set of functions and UIs compared with the original software.

BACKEND CHANGES

Since our software is implemented and structured in OSGi bundles, it is relatively easy to add, replace or remove functionality by deploying or removing bundles. So adding "white label" capabilities to our system was not really a technical challenge since the underlying architecture directly supports the necessary configurability.

To extend the backed for OEM use, we had to extend our system database for multi-client capabilities. This was done by extending the data model to include contract data for the OEM customers. The contract types are being used to configure which part of the software e.g. which OSGi bundle is included in the runtime environment for which particular contract.

The customer can also order (or implement) some kind of extra functionality besides the preexisting bundles and include them into his contract configuration.

In addition it was also necessary to provide a way to set specific contract information for any particular EnergyBASE device and to integrate the dynamically configured extensions into the systems management. mPRM provides an appropriate technology (called "control units") which we used to monitor custom extensions on the backed.

>>

Illustration 7: UI effects by changing contract

DEVICE CHANGES

As a part of the implementation of the EnergyBASE device software, we developed a possibility to mark a bundle within its "Manifest.mf" configuration file as "ManagedByContract". Such bundle will only be loaded when the currently applicable contract calls for it. The contract information is managed by a software component called ContractService. This service receives every change in the contract from the backend instantly through an event system and begins to start/stop different bundles according to the new configuration.

Additionally, we did many changes to the local UI layer and separated these bundles into smaller pieces. Now onwards, there is one bundle with the standard UI and one additional for each customer. This customer bundle does not only contains UI stuff but also arbitrary code to solve the required features or change existing behavior. In the case of UI, it contains just the difference between standard and customer specific CSS/HTML/Assets. The Illustration 7 demonstrates this kind of change in Web-UI just by changing the contract. Everything happens without any reboot or manual browser refresh.

Due to the dependencies between bundles shown in Illustration 4 it is possible to imagine that there can be several tricky situations to think about while developing this part.

CONCLUSION

The Java/OSGi software platform used for the EnergyBASE project has proven to be stable, flexible and extensible. Homogeneous runtime environments on all system components allow us to distribute code and functionality as needed. In particular, the OSGi component model and it's hot deploy/undeploy capabilities helped us to quickly implement the customer requirement to expand the system from a proprietary offering into an open multi-client platform.

Even new requirements, like opening up the system as a hosting platform for domain specific third party apps would not be a significant challenge for this robust architecture.

THE FIRST OEM CUSTOMER

Let's have a quick look on the requirements of our first real world OEM customer and the resulting efforts in development.

1. The EnergyBASE web app should not be available anymore after the installation is done.

   => Easy. The UI bundle will be marked as "ManagedByContract" and not included in the related contract-configuration.

2. The customer wants to develop its own mobile application which displays the data collected by the EnergyBASE every five minutes.

   => Just implement a mechanism to send the needed data every five minutes to the mPRM within the customer-specific bundle. A kind of observer mechanism will be triggered after receiving the data on the mPRM side. After that, just forward it to the customer.

3. Based on some cooperation contract, only two manufacturers should be available in this configuration. A specific solar inverter and one type of battery. Other devices and manufacturer should not be supported.

   => The same procedure as we did in point 1. The device adapters are already split into separate bundles for each manufacturer. Just add the allowed devices in related configuration.

4. The remote support access should be activated all the time.

   => The original Version of EnergyBASE software does not force the user to enable the option for remote access. In this case, we just have to override this option within the customer-specific bundle.

5. Additional function: Each device should consume and store the stock prices for electricity by using a defined service once per day.

   => Create a new "ManagedByContract" bundle and add it to the contract configuration.

6. Additional function: The customer should consume energy for free to load its battery in the time periods with negative prices (based on the data from the previous point) until a specific amount is reached.

   => The same procedure as described in point five. This real-world example clearly shows that significant deviations between customer requirements and default implementation can be managed in a very clean way. We don't need complex and error prone if/then/else code to solve the challenge.

   A split of functionality and responsibility on our bundles and the overall modular approach help us manage most of the requirements with a very low effort on the development side. All complex enhancements were separated in custom bundles without any need to change the existing code.

Refrences:

EnBW Energie Baden-Württemberg AG – https://enbw.com
EnergyBASE – https://energybase.com
Oracle Java SE 8 Embedded – https://oracle.com/technetwork/java/embedded
ProSyst mBS Smart Home - https://dz.prosyst.com/pdoc/mBS_SDK_7.5
OSGi – https://www.osgi.org/

FAKE ?

My first encounter with counterfeit components occurred more than 20 years ago while working for a large North-American electronics manufacturer. One day a fellow test engineer was called to the production line to help investigate the 100% test failure of a certain product. We headed down to the line, and quickly isolated the failures to a specific component. Getting no electric response from the part, we decided to X-ray it to look for damage or broken wire bonds. Looking at the images in disbelief, we had to check several times before accepting that the component housing contained neither chip nor wire bonds. Our boards were all populated with empty capsules marked, labelled, packaged and passed off as real components.

BY: Haldor Husby
Principal Development Engineer
Data Respons

In hindsight, the innocence of our shock facing such simple component deception seems almost quaint. We were stunned by the sheer audacity of the fraudsters and did not realize that hawking empty capsules as real components is actually one of the more benign forms of counterfeiting. Counterfeit electronic components, at that time an almost unknown issue, would accelerate in the 10 years following to become a visible and acknowledged problem with thousands of reported incidents in 2005 which in turn increased another 300% by 2008.

The issue of counterfeit components pivoting around the turn of the millennium is closely related to fundamental changes in the electronics component supply chain at and around that time. The admittance of China into the World trade Organization (WTO) in 2001 resulted in the lifting on export bans for non-governmental entities. A surge of manufacturing outsourcing and the development of global shipping shifted the manufacturing center of gravity to Asia, specifically China, a region with weak protection and understanding of intellectual property, creating distance between the OEMS and their supply chain. Somewhat earlier, major efforts to establish a responsible E-waste handling led to a massive export of hazardous waste in the form of discarded electronics to China and other developing countries, creating a substantial industry centered on e-waste recycling. This industry, intended for the recovery of precious metals from electronic assemblies, became a growing source of reclaimed electronic components.

TYPES OF COUNTERFEITS

The word counterfeit invokes associations of unauthorized copies. An early and famous case affecting thousands of computer motherboards involved a capacitor electrolyte made from a formula first stolen, then corrupted, which caused the capacitors to burst and the computers to malfunction. The case alone cost the computer makers more than USD 100 million. However, making copies, now specifically termed cloning, is just one of many ways of creating counterfeit parts, and not even the most common. Other major sources of counterfeits are excess inventory improperly disposed of, legitimately produced parts rejected by the test process, legitimate parts remarked and re-labelled as parts of better performance and the aforementioned empty capsules.

But the most common, and perhaps most sinister counterfeits are parts reclaimed from used and discarded electronic products, primarily in Chinese backyard operations. The boards are typically heated over open fire to as much as 400°C (far higher than the approved rated reflow temperature) to liquefy the solder, then hit and thumped to the concrete floor until the parts fall off. After sorting and cleaning in whatever water is available at the site, the top markings are ground down and a new topcoat is applied before the parts are marked, labelled, packaged, and reintroduced as fresh parts through the grey market.

This group constituted an estimated 80-90% of the component counterfeit market in 2012, which in turn was assumed to be 8-10% of the total electronic components market and representing an annual revenue loss of USD 7-8 billion to the semiconductor industry. However, this is only a fraction of the overall cost counterfeit components represents to society, albeit maybe the only one that is close to quantifiable. Correcting a problem invoked by a counterfeit component, once detected, may exceed the value of the components by orders of magnitude. A counterfeit component not detected may cause serious loss of infrastructure in the worst case, and the loss of life and safety for people.

> "These hard-to-come-by parts are most easily found in the grey market.

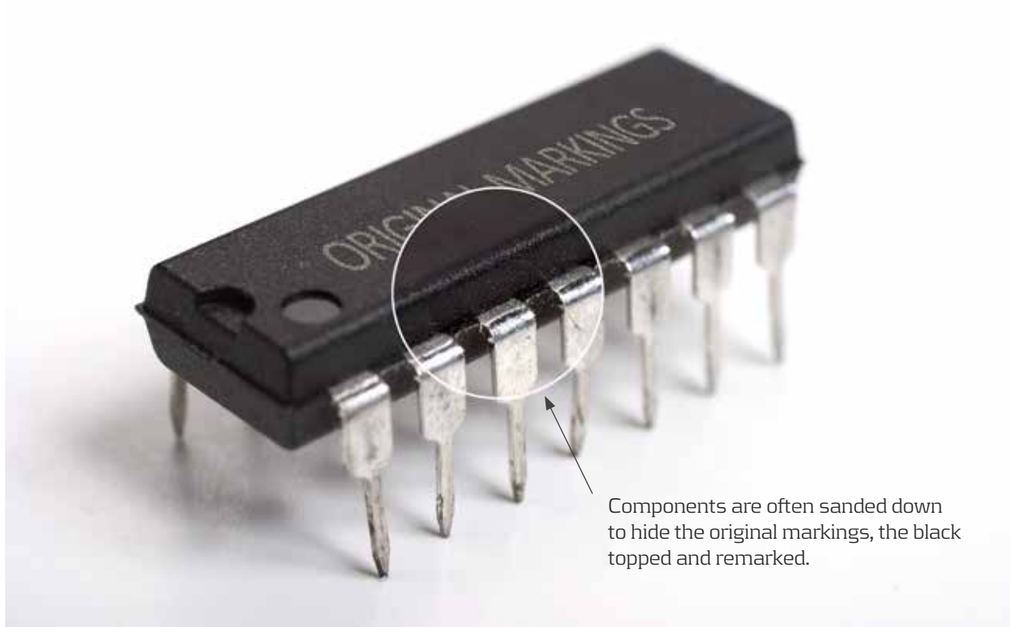
RISKS AND CONSEQUENCES

A salvaged waste component already spent an unknown and possibly significant percentage of its useful service life before being recycled. Add to that an unknown, and possibly inappropriate service situation, and the fact that the part is on a board that has been discarded, and it is clear that re-claimed electronic components can only be legitimately used in non-critical applications. However, the vast majority of reported counterfeit incidents are in the military and aerospace segments, and includes cases involving safety and mission critical systems. It is clear that these segments are particularly susceptible to counterfeits, and not only discovering incidents to much higher degree. Since the 2011 reported discovery of counterfeits in major military systems like the F16 fighter jet, and the realization of the risks it represents, the attention on fighting counterfeits has been intense, far greater than in regular commercial markets. The test and qualification regimes of the defense sector along with the consciousness of the consequences of failure contributes to a superior detection of substandard quality. However, defense and aerospace, with product lifetimes spanning several decades, are especially mismatched to components life cycles of a few years, and do rely on a steady supply of components that are in effect obsolete. These hard-to-come-by parts are most easily found in the grey market. Hence, the problem of counterfeit components are closely related to the ever-mounting problem of obsolescence and life-cycle management (see article in the previous issue of Interrupt Inside).

On the face of it, avoiding counterfeit components should be simple; buy components directly from component makers and reputable authorized distributers only, and you have no problem. Not until you need a part not available through those channels, that is. Considering a complex military system like a fighter jet or a helicopter, adding to it variants, upgrades and maintenance, it is obvious that the supply chain is extremely large and convoluted involving sub-contractors with sub-contractors at multiple levels. Each of them battling their own difficulties with obsolescence, lead times, delivery pressure and cost, and with varying levels of maturity and control handling the parts supply, not to mention ethics. The temptations to make use of the grey market are multifold. A vendor of a sub-component pressured and committed to a delivery date but missing a handful of critical components, gets instant relief from a smaller broker. An EMS provider, cut to the bone on price by his customer, sees the opportunity of recovering some of his profit procuring the most expensive parts from another friendly broker. The appearance of counterfeit components in military planes is no mystery, once you know it.

FIGHTING COUNTERFEITS

Early topcoats could easily be removed with an Acetone wipe, and date and lot codes printed on the components themselves were often incorrectly formatted relative to the specifications from the vendor. Fakes were therefore relatively



Components are often sanded down to hide the original markings, the black topped and remarked.

DEPARTMENT OF DEFENSE DEFINITION

An unlawful or unauthorized reproduction, substitution, or alteration that has been knowingly mismarked, misidentified, or otherwise misrepresented to be an authentic, unmodified electronic part from the original manufacturer, or a source with the express written authority of the original manufacturer or current design activity, including an authorized aftermarket manufacturer. Unlawful or unauthorized substitution includes used electronic parts represented as new, or the false identification of grade, serial number, lot number, date code, or performance characteristics."

SAE INTERNATIONAL STANDARD AS5553 DEFINITION

A fraudulent part that has been confirmed to be a copy, imitation, or substitute that has been represented, identified, or marked as genuine, and/or altered by a source without legal right with intent to mislead, deceive, or defraud.

easy to detect once looked for. However, counterfeiters are steadily getting better at what they do, so the technology to detect frauds must improve as well. Companies are, as an example, working on using botanical DNA to mark chips, and the use of RFID tags has long been considered, but the long term impact of improved marking will only be relevant for cloned components. To aid detection, IPC has developed inspection training and certification for detection of counterfeit components. One must assume that frequent re-education is necessary. Several automated test and inspection systems targeting detection of counterfeit components are also about to hit high-end markets.

Parallel to improvements in process and technology, the distribution of counterfeit components is fought in American courtrooms. In response to relatively resent definitions of new offences introduced in law motivated by the appearance of counterfeit parts in defense systems, the FBI has stepped up its investigation of component fraud, and several American brokers have been subject to high-profile prosecution and sentenced to lengthy incarceration. The message is clear; if you supply US defense companies you need to be sure that all components are genuine, or face charges and prison terms. The original "manufacturers" and distributers of the counterfeit parts are of course still out of reach.

Inspection and prosecution aside, getting on top of the counterfeit component problem necessitates getting in control of the supply chain. Traceability from manufacture through distribution and assembly is inevitable for any OEM or sub-system manufacturer who wants to be confident that their product is clean of counterfeits. Also in this context, coordinated industry responses are important. Like the SAE internationals standard AS5553 for procurement of electronic parts, and directly motivated by volume of fraudulent parts in the supply chain.

STAYING SAFE

Counterfeit products are not limited to electronic components. Fakes, primarily clones, are widespread in all markets. Every year more than a 100 million fake phones are put in circulation. Fake ball bearings, car parts, cables, network servers, safety textiles, vehicle airbags and many more are well known and severe examples of counterfeits discovered. Considering the profits involved, the fragmentation of the supply chain and the many pressures on manufacturers, there is little reason to expect the fight to end counterfeiting to be successful. It appears that the only path to successful avoidance of counterfeit components goes through solid life cycle management. The link between counterfeit component avoidance and obsolescence management cannot be overstated, and actions taken to avoid "distress procurement" are also actions to keep fake components out of the factory.

SYSTEM MODELING

BY: Fredrik Bakke
Senior Development Engineer
Data Respons

BY: Svein Tore Ekre
Senior Development Engineer
Data Respons

The concept of modeling system requirements and design in is not a new one. However, recent advances in languages and tools has created opportunities for reducing the total development effort for embedded systems, and improve quality. This article aims to present some of these opportunities, based on the authors' experiences. Keywords are traceability, and multiple consistent requirements, design, and test views. The article gives an overview of the SysML language, its usage, and potential benefits. The article also gives advice on how to get started with system modeling along with literature recommendations.

SYSML BACKGROUND

The Block diagrams and flow diagrams are perhaps the de facto standards of visually describing structure and functions of embedded systems. These diagrams have good track records, but one shortcoming lies in the fact that they can not easily be integrated with other diagram types or even other diagrams of their own type using common tools. If you for example develop five block diagrams in Microsoft Visio with some common elements, you will need to maintain each of them individually. Make a change in one, and you will need to validate the others. If flow diagrams for these elements are also developed, the effort of making changes and ensuring consistency increases exponentially. A solution to making multiple diagrams consistent, is to use a tool that integrates several diagrams and diagram types using a relational database. This was done by UML to unify the world of software modeling. UML is now a 20 year old mature software modeling language that promotes an object oriented mindset. A major strength of UML is the ability to combine diagrams showing SW Structure and SW behavior, and reuse elements. Reuse is also a key aspect of SysML, and portions of models can be reused between product generations or variants.

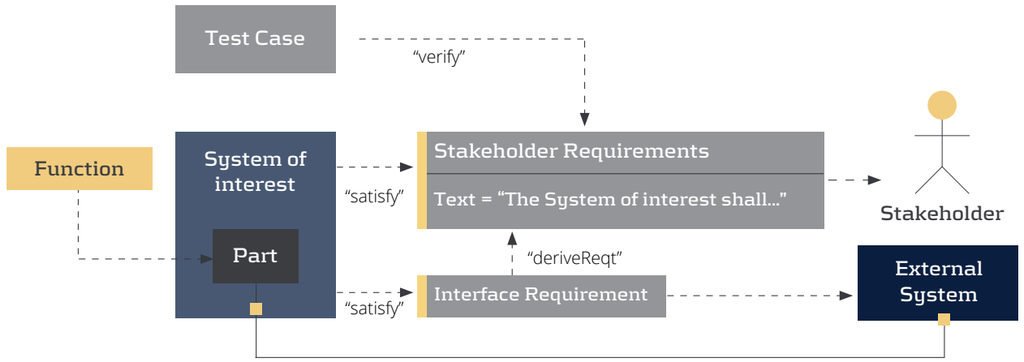
The most used UML diagram types are arguably the class, sequence, and use-case diagram. UML tools lets elements be reused between diagrams, and a change in one element is therefore reflected in all diagrams showing that element. This enables more aspects of a system to be documented with diagrams, for less effort. Common criticism of UML is that its "red tape" that gets in the way of coding, and UML will indeed let you describe SW all the way down to SW function level. Compared to UML, the Systems Modeling Language (SysML) is more light-weight, more general, and targeted towards modeling requirements and architecture.

In 2001, the International Consortium on Systems Engineering (INCOSE) and the Object Management Group (OMG) issued the "UML for Systems Engineering" request for proposal, with the intention of adapting UML for system specification and design. In the 16 years since, SysML (now at version 1.4) has developed into a mature and more agile language than UML, that is suitable for modeling requirements, hardware, software and processes. In addition, a SysML model gives opportunities for documenting the relationships between requirements and system components at any level of decomposition in accordance with best practices and also functional safety requirements.

SYSML 1.4 AND ENTERPRISE ARCHITECT

The SysML language is a profile of UML, and provides both a notation in the form of diagrams, elements, and relationships, and the semantics of these. Some diagrams are directly adopted from UML, the requirements diagram and parametric diagram are new diagram types, and some UML diagrams have been left out in SysML. The authors have used Enterprise Architect from Sparx Systems for SysML modeling. It is a feature rich and flexible modeling tool with good Support for SysML 1.4.

The process of modeling an aspect of the system is to first create a diagram of a suitable type (see SysML 1.4 Diagram Types), secondly drag in any previously defined elements, and third define any



new elements or relationships. After that, descriptive text and visual formatting can be added for increased readability. Enterprise Architect lets you hide relationships and element properties on a per diagram basis, so a diagram can show what you want and nothing more.

SYSTEMS MODELING IN DATA RESPONS

SysML models have been used in Data Respons by the authors since 2011. The applications have ranged from concept studies, internal process descriptions, through requirements specifications and architecture descriptions. For requirements specifications, the authors have created model with full bi-directional traceability between system requirements and environmental requirements (Aviation). In the automotive industry, the authors have established bi-directional traceability in a system requirements specification down to software unit level. This has shown that SysML models can be efficient means to achieve traceability between system requirements and stakeholder requirements, and also down to low level design.

The authors have also used a SysML model for stakeholder management, capturing requirements, exploring solution concepts and developing system architecture. Capturing this information in the same model has shown the strength of using a model containing diagrams as a tool for communicating and validating design decisions in an iterative manner. The model also proved to be efficient for establishing a shared terminology and understanding of the system under development, for exploring solution concepts in team, and for documenting system architecture at multiple levels of decomposition.

MODELING TOOLS

There are some SysML capable tools to choose from, both with commercial and open source licenses. Googling "SysML tools" yields lists of popular tools, comparisons and feature lists. SysML underwent significant changes up till version 1.3. The current version of the standard (1.4) has been around since 2015. Some tools are better than others in implementing new SysML features, and not all tools available have mature enough SysML support for efficient system modeling. No Magic MagicDraw, Altova Umodel and Sparx Enterprise Architect are among the most popular SysML capable modeling tools.

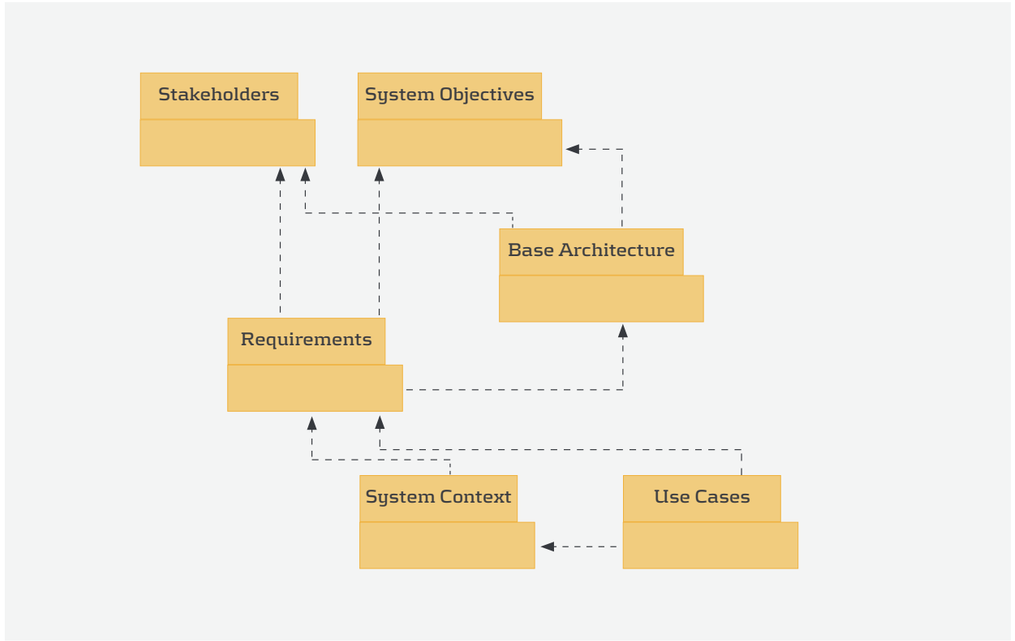
SysML 1.4 Diagram Types	
Diagram type	Usage
Block Definition Diagram	Define reusable structure and behavioral elements
Internal Block Diagram	Connections, interface, internal block structure
Activity Diagram	Flow based behavior
Sequence Diagram	Message based behavior (Communication protocols)
State Machine Diagram	State based behavior
Use Case Diagram	Stakeholders and use-cases
Requirements Diagram	Requirements and Test Cases
Parametric diagrams	Quantitative Constraints
Package Diagram	The structure of the model, and relationships between packages

GETTING STARTED

SysML is a language. In order to create a model in the SysML language that serves a purpose in a given project, the purpose must first be defined, this might for example be a requirements specification, consistent design diagrams, interface specifications, test management, or a full architecture description. Secondly, a suitable model structure and workflow must be established. This is arguably the most critical challenge of working with SysML models. Knowledge of Systems Engineering best practices and experience with SysML or UML and the modeling tool is recommended. A SysML model is structured using packages. These are logical containers that contains diagrams and other elements. Elements in the same or different packages can have relationships between each other. Even though the element and relationship types have defined SysML semantics, practice shows this is not always clearly enough defined. While using the semantics of the SysML standard is a good thing, we have found that the usage of diagram, elements and relationship types should not be too tied to the SysML semantics, but instead be documented on a per-model basis. Thus ensuring consistency within the model. The package diagram below shows one possible model structure with package dependencies.

MODEL SETUP RECOMMENDATIONS

SysML models should be structured on a per project basis in order to meet project specific requirement. However, the experience of the authors is that following some general rules when setting up the model results in it being more readable and maintainable.



- Diagram Legends can be defined once and used in several diagrams. Use Legends and Element colors consistently to help make the model readable.
- All diagrams should have a text box, describing what aspect of the system the diagram shows. For improved readability, do not rely solely on SysML notations like the diagram header.
- Document the usage of SysML element and connector "Types" and "Stereotypes", and make sure the meaning is unique. This is a prerequisite for a consistent model, improves traceability, and enables complex model searches.
- Use a "package" diagram to establish a package structure and document package dependencies. This serves as an overview of the model, and helps in managing changes to the model.
- Define a logical system break-down, and structure all information in accordance with this. The items in the breakdown structure should represent logical parts of the system (housing, power module, controller SW etc). Break down as many levels as needed.
- Manage the model scope, and stop to consider return on investment before modeling below "architecture level".
- Quick visualization of relationships benefits from a properly structured model. Make sure you understand the visualization capabilities of the modeling tool before deciding on model structure. For Enterprise Architect this is the "Traceability view", the "Relationship Matrix", and "Insert Related Elements" feature.'

> The visual notation of SysML gives the model user a quicker understanding of requirements and architecture

TRACEABILITY, REPORTING AND VISUALIZATION

Projects may require traceability between stakeholder requirements, system requirements, components, test-cases and tests at different levels. As long as the "Model Setup Recommendations" are followed, custom searches can be saved and performed quickly on the model without the need for documenting complex relationships directly. Examples of possible custom searches are listen below:

- Passed test-cases at SW component level tracing to a set of stakeholder requirements.
- Components impacted by changes in a requirement, and tests that must be re-run.
- List of stakeholder requirements not yet verified at component level.

Sometimes, there are better ways to present or share information than using diagrams. Most SysML capable modeling tools have several options for reporting and presenting data. Enterprise Architect has a customizable report generator for MS Word and PDF, .html generator and an .XMI import/export function in addition to version control integration. Also, relationships in packages can be presented using the relationship matrix, and any diagram can be presented on list format. This makes it possible to generate different but con-

sistent reports and visualizations.

SYSMOD - THE SYSTEMS MODELING TOOLBOX

SysML provides a language with notation and semantics, but do not advise on the process of system modeling. SysMod is a framework for modeling the system from stakeholder requirements to a product architecture. SysMod uses examples with SysML and Enterprise Architect. This can be a good starting point for determining the scope of the modeling effort and model structure. SysMod describes at a high level what should be modeled and the relationship between packages. See Literature Recommendations for a description of SysMod.

Example SysMod Products
System Objectives - The high level objectives of the System under development
Stakeholders - Anyone or anything that has and interest in or interacts with the system during its lifecycle
Base Architecture - Design decisions made before the project started
Requirements - Stakeholder and System Requirements and Constraints.
System Context - External systems, operators, environmental effects etc.
Logical Architecture - The System Architecture, reusable between product variants
Product Architecture - Product variant/generation specific architecture

SUMMARY

SysML with supporting tools provides opportunities for reducing the documentation effort and increase quality in all stages of development projects. In the initial phases of a project, a SysML model can improve communication and help validate requirements and design decision. Inconsistency will be easier to discover by the use of visual models.

The project manager can track progress using custom searches across complex relationships. For example by the number of customer requirements that are verified at component level. After traceability between requirements, design and test has been established, use-cases or user stories can be prioritized for each phase of development more efficiently. Impact analysis diagrams can be generated and used for change management. Use case or user-story based development can benefit from giving the developer auto generated views of requirements and architecture. This provides relevant information for the specific use case or user story.

The ability to document relationships from system objective, through requirements and design makes it possible to trace all functionality back to customer requirements and business value. This is also valuable for testing, as test coverage can be easily measured. Diagrams showing complex relationships can be auto generated based on custom searches.

At project delivery, customer documentation or internal documents can be auto generated from the model, using custom templates. For example design descriptions, interface descriptions and test reports. Consistency is ensured when all reports are generated from the same model. The model can also be reused for future generations of the product to speed up the initial phases of a project.

If a model is structured in a manner that facilitates its purpose, the results can be requirements and architecture descriptions that are more consistent and less time consuming to develop and maintain than document based specifications. The visual notation of SysML gives the model user a quicker understanding of requirements and architecture, this can make collaboration with stakeholders and within the development team more efficient. A prerequisite for this is that some modeling guidelines are followed in structuring and developing the model. The model structure decided on initially will impact its usability later on in the project. Consideration of the models' purpose and potential scope must therefore be given as early as possible. This article gives some recommendations for structuring system models. To get a clearer picture of the opportunities and limitations of SysML models and the Enterprise Architect modeling tool, we recommend the literature listed at the end of this article. SYSMOD - The System Modeling toolbox gives an overview of the modeling process, and can be great input for deciding on the modeling scope and structure.

Literature Recommendations

A Practical Guide to SysML
Description: The SysML Language
Author: Friedenthal, Moore, Steiner

SYSMOD - The Systems Modeling Toolbox
Description: A Systems Engineering process based on best practices, that uses SysML.
Author: Tim Weilkiens

50 Enterprise Architect Tricks
Description: Useful tips and tricks for modeling in Sparx Enterprise Architect.
Author: Peter Doomen

EA in 10 days
Description: Introduction to Sparx Enterprise Architect.
Author: Peter Doomen

References

INCOSE - International Council on Systems Engineering
http://www.incose.org/about

UML - Unified Modeling Language
http://www.uml.org/what-is-uml.htm

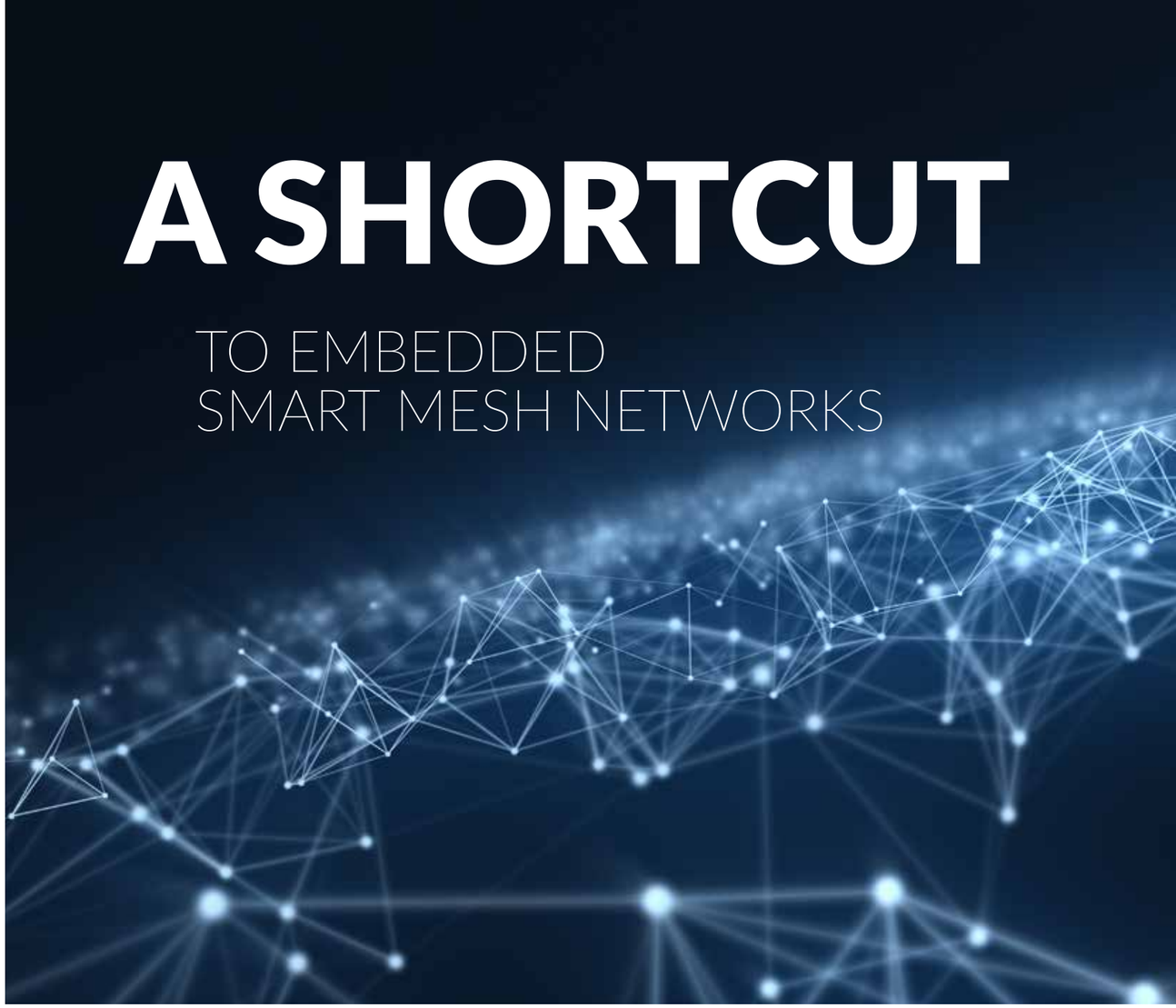
SysML
http://www.omgsysml.org

OMG
http://www.omg.org

SysMOD
https://leanpub.com/sysmod

A SHORTCUT

TO EMBEDDED SMART MESH NETWORKS



In the emerging world of Internet-of-Things, wireless low-power mesh networks are more relevant than ever. Data Respons has gained valuable experience with one particular technology after employing it in a large industrial instrumentation project, namely Linear Technology's SmartMesh IP. As a specialist on embedded solutions, Data Respons recently became an official Linear SmartMesh partner, after developing the QuickStart Library: A software library that greatly reduce development time for embedded applications of SmartMesh IP.

BY: Jon-Håkon Bøe Røli
Development Engineer
Data Respons

MESH-TO-THE-EDGE

SmartMesh IP is a wireless technology pioneered by the Linear Technology owned company Dust Networks. A descendant of ultra-low power and ultra-high reliability protocols such as WirelessHART, the SmartMesh IP protocol is based on the 6LoWPAN and 802.15.4e standards. It features a time slotted, channel hopping mesh network where every node knows exactly when to listen, talk or sleep, resulting in a very power efficient and collision-free packet exchange. Every device in the mesh network has the same routing capabilities, often referred to as "mesh-to-the-edge", as it provides redundant routing to the edge of the network. This allows for a self-forming and self-healing network that constantly adapts to changes in topology, while maintaining an extremely high data reliability, even in harsh radio frequency environments.

MOTES AND MANAGER

A SmartMesh IP network consists of one or several wireless nodes, known as motes, which collect and relay data, and a network manager. The manager has two fundamental functions: Firstly, it is an access point (AP) that acts as a gateway between the mesh network and the monitoring or control network. Secondly, it runs the network application software that continuously makes decisions on how to build and maintain the mesh network.

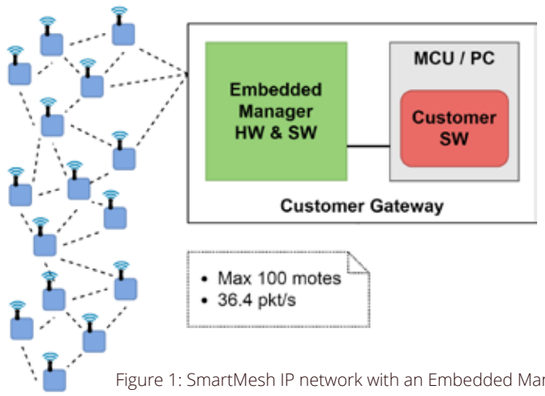


Figure 1: SmartMesh IP network with an Embedded Manager.

The Embedded Manager is a self-contained solution where both the AP function and the network management algorithms runs on a single chip. This setup however, illustrated in Figure 1, is limited for smaller networks, as the single AP has a hardware constraint of 100 motes and a throughput of 36.4 packets per second. The customer software communicates with the manager directly through a serial Application Programming Interface (API).

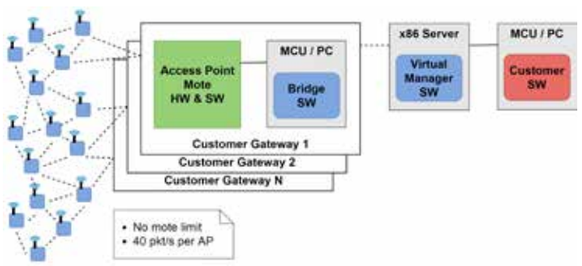


Figure 2: SmartMesh IP network with a Virtual Manager.

A second, new alternative is the Virtual Manager, where the network application runs on an x86 virtual machine, while only the AP functionality remains on-chip. The AP, together with a bridge SW on a locally connected MCU or PC, then constitutes an AP gateway that connects remotely to the virtual manager. This connection can be serial, Ethernet, WiFi or even cellular, as long as it can support the maximum throughput of 40 packets per second from the AP. In this setup, illustrated in Figure 2, the customer application interacts with the virtual manager through an HTTP-based API. Adding multiple APs can scale the network to support thousands of motes, as well as increase the available throughput, reduce latency or achieve redundancy.

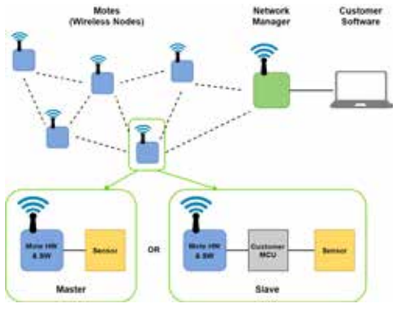


Figure 3:
Master or Slave - The two modes of a SmartMesh IP mote.

MASTER AND SLAVE

The typical use for a mesh network is to publish sensor data from each node to a centralized application for processing, storage and/or visualization. As illustrated in Figure 3, a SmartMesh IP mote can operate in two different modes. Running in a master mode, the on-board ARM Cortex-M3 processor can access sensors and other I/O directly, where it runs an application that terminates commands and controls network joining. An On-Chip Software Development Kit (SDK) allows a user to write applications directly on the mote, on top of the SmartMesh IP network protocol stack. Alternatively, the mote can run as a slave to a connected device, expecting the master device to terminate commands and control network joining via a serial API. This puts more complexity in the hands of the user, but is often the most viable option in an embedded solution, as a custom MCU adds more flexibility.

C LIBRARY

Since both the SmartMesh mote and embedded manager has a similar serial API that the typical embedded application has to interact with, Linear provides a complete implementation of both in the SmartMesh C Library. This library abstracts commands into simple function calls, handling serial formatting and framing for the high-level data link control (HDLC) protocol used in all serial communication with SmartMesh devices. The library also makes sure to match sent commands with ensuing replies, passing them back through a callback function. Notifications received from the SmartMesh device are also parsed and correctly acknowledged, before they too are passed back "up" through a callback function. Still, implementing the API itself is not necessarily the hardest part. On the manager-side there is little to no required intervention, as it will autonomously start creating a network upon power-up – The connected customer software simply need subscribe to the desired notifications, while commands and interactions are stateless, and thus reasonably straightforward. By contrast, on the mote-side a software designer has to be aware of mote states and corresponding behavior, as well as the correct sequence of configurations and commands to join a network. Linear found that this knowledge barrier sometimes prevented potential customers from embedding SmartMesh IP in their applications, which is why the need for a simpler starting point emerged.

QUICKSTART LIBRARY

The QuickStart Library (QSL) developed by Data Respons abstracts the mote interface one step further: A finite state machine (FSM) schedules the necessary sequence of commands depending on the current state, events and replies from the mote, leaving only a minimal and intuitive API for the user. For example, the steps necessary to configure the mote, set up sockets, initialize a search for and join a network, as well as request a certain bandwidth, are all hidden in a simple call to **connect**. Downstream user payloads are also handled by storing them in a circular inbox buffer with a configurable size, where calls to **read** will pop the oldest message in the inbox, if any. **Send** queues a payload for transmission to the manager, while **is Connected** is a simple way to check if the mote is still connected (this way the user application can determine if a failed **send** is the result of not being connected or an actual transmission failure). Lastly, **init** should be called once upon startup, and will simply initialize the data structures and establish the serial connection to the mote. Except for **read**, which returns the number of bytes read, the API only returns simple Booleans to let the user application know if an attempt was successful, avoiding the need to interpret any response codes. Furthermore, **send** and **connect** has a configurable timeout such that the user application can be sure that their call will return within a set limit. While **send** only makes one attempt at queueing a packet, **connect** keeps trying to join a network until successful or the configured timeout occurs.

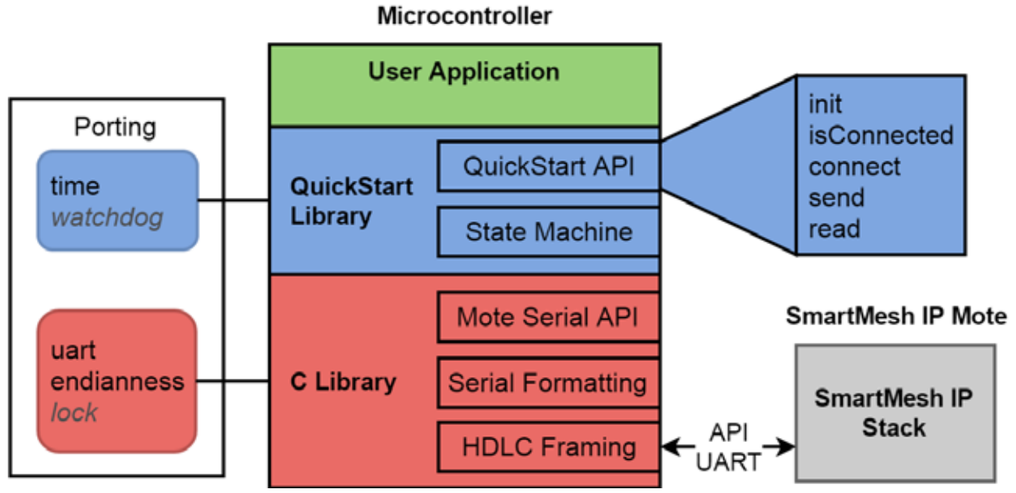


Figure 4: The QuickStart Library structure.

PLATFORM INDEPENDENT

Designed to be highly portable, the QSL (and the underlying C Library) is written in C without any hardware-specific code, allowing its use "as-is" in any C-based platform. Platform dependent functions only have their prototypes implemented, leaving their definition to the user. For instance, a developer has to define how to feed the watchdog (if any) or how individual bytes are written to or read from the serial port. Figure 4 illustrates the library structure, where the hardware-specific categories that need definitions are listed to the left (watchdog and lock for concurrency is optional).

To further help developers get started, complete sample code is provided for a set of commonly used platforms: Raspberry Pi, Atmel SAM C21 and STM32, as well as a generic example for the ARM mbed operating system. Sample code for these platforms also include implementations of the necessary prototypes.

RAPID MESH NETWORK PROTOTYPING

QSL is accompanied by a detailed guide, with step-by-step instructions on how to get started with the typical case of data publishing from an external MCU. The guide also explains how to get a demo up and running with the sample code provided for the supported platforms. It also includes guidance on existing tools that can visualize data arriving on the manager-side, as well as transmit data downstream to motes. This allows a developer to integrate a prototype mesh network with their embedded system within only a few hours.

As the name entails, the QSL is primarily meant to help developers get started with embedding SmartMesh IP in their applications. The library is not an exhaustive API for the SmartMesh IP mote, although its interface is adequate for most simple applications, as it provides functionality for data transmission and configuration of the most important network settings. Furthermore, by extending its functionality or simply by using it as a thorough how-to, the QSL can reduce development time for advanced applications that require more features from the underlying mote interface.

> "The steps necessary to configure the mote, set up sockets, initialize a search for and join a network, as well as request a certain bandwidth, are all hidden in a simple call to **connect**.

Note:
On March 10th, 2017 Linear Technology Corporation officially became part of Analog Devices. Inc.

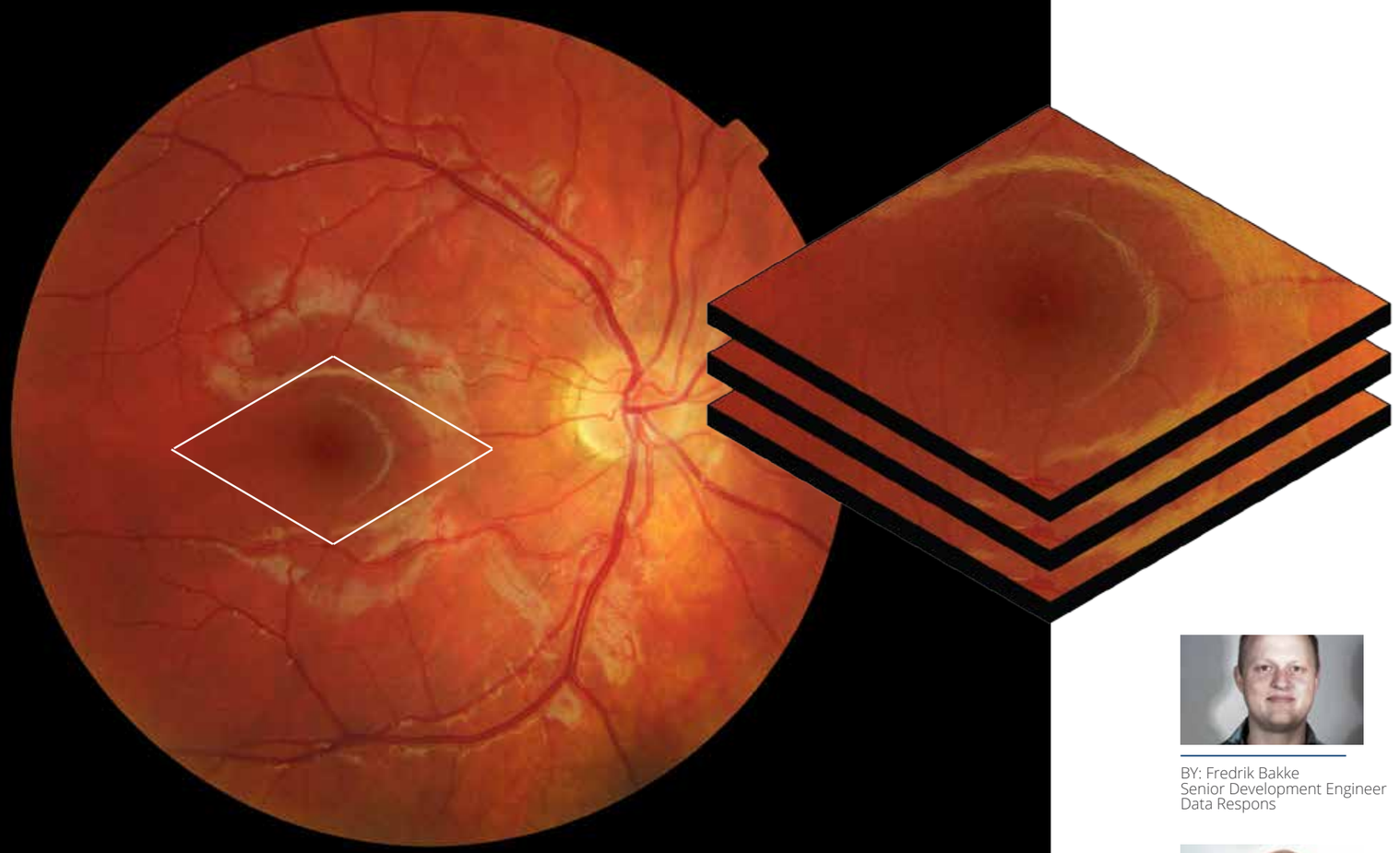


Image classification

using artificial neural networks

BY: Fredrik Bakke
Senior Development Engineer
Data Respons

Co-author: Lars Albert Fleischer
Senior Development Engineer
Data Respons

Co-author: Svein Tore Ekre
Senior Development Engineer
Data Respons

Co-author: Eimund Strøm
Specialist Development Engineer
Data Respons

Machine learning using neural networks is a field that has seen rapid development the recent years. While the basic theory of artificial neurons linked together in Artificial Neural Networks (ANN's) was developed in the fourties, the hardware for efficient implementation of relatively large ANN's has only recently become commoditized. In the wake of these advances, frameworks for building and training ANN's have been developed - many with open source licenses. Unlike in traditional computing, neural networks can be designed and trained to identify complex patterns without specifying the pattern-features in an algorithmic manner with if's and else's. Image Classification is one of the fields where neural networks can be put to use.

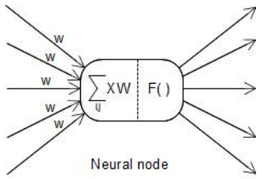
In 2016 Data Respons was asked to collaborate with the University College of South-east Norway (HSN) in a research project with the long term goal of applying machine learning and image classification to diagnose the eye disease Age-related Macular Degeneration (AMD). The project has financial support from Oslofjordfondet, a fund that promotes innovation and R&D in the region around Oslofjorden. The picture (at location) shows the backside of the eye interior (fundus). The marked area shows a drusen, which is a buildup of fatty protein. The shape of the drusen shows it consists of more than one smaller drusen, and can therefore be a sign of AMD. This is the feature the ANN-based classifier needs to detect. This article aims to give an introduction to the theory and available tools for building and training image classifiers based on neural networks, with root in experiences from the AMD diagnosis project.

CONVOLUTIONAL NEURAL NETWORKS (CNN'S)

The concept of artificial neural networks is inspired by nature. Biological neural networks consists of cells called neurons that are connected with synapses. Over the synapses, the neurons fire electrical signals. Similarly, Aritificial Neural Networks consists of nodes transmitting values to the input of other nodes. Two of the factors that separates biological and artificial neural networks is scale and complexity. An average adult human brain has 100 billions neurons each with 7000 connections to other

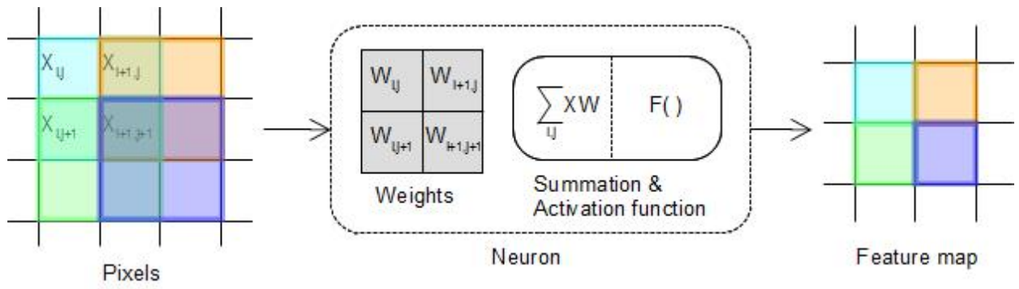
neurons. This is currently outside the capabilities of modern hardware.

The image below shows an example of a single artificial neuron (node) with input and output connections. Each input connection has a weight constant that is used to define the importance of the connection. The inputs of the neural node are multiplied with the weights, summed and then passed through an activation function. The output from the activation function is usually between 0 and 1.



A network architecture commonly used in image classification is the Convolutional Neural Network (CNN). CNN's are feed forward networks that are inspired by the visual cortex. A CNN consist of a few distinct types of layers, convolutional layers, pooling layers and fully-connected layers. In a feed forward network, the nodes are divided into hierarchical layers where each node only transmit to a node at the level above.

The image below shows an example of the convolutional layer in a CNN. Starting with a 2x2 pixel group from an image, the pixel group is multiplied with a 2x2 filter weight, summed and then passed through an activation function. The output from the activation function is called a feature. The weights are then shifted and applied to another four pixels to calculate another feature. Features calculated for all the pixels in the input image then generate a feature map. More weights can be used to generate more feature maps.



The pooling layer of a CNN do a subsampling to reduce the amount of features and the computational complexity of the network. The most commonly used pooling layer is the maxpool layer, which pick the largest element from a group of features. Typically this group is a 2x2 window and thereby reduce the amount of features by 75%. The main reason for the pooling layer is to prevent overfitting, which is when the network describes random noise instead of the general trend in the data. An overview of the layers is shown in the figure at the next page.

The fully-connected layer perform the high level reasoning in the CNN in order to output the final scoring.

Designing a CNN involves a number of parameters whose effect on accuracy is not necessarily intuitive. Trial and error is therefore a large part of creating a CNN, and this make the training process time-consuming. The design often start with pre-processing of the images, to reduce noise and bring out the features belonging to the classes of interest.

The next step is to design the network. Decisions that impact the accuracy of the network include how the image is divided among nodes, and the pixel group size and overlap best suited to detect the required features. The final step is the amount and type of layers and connections. Generally, more layers enables classification based on more complex patterns.

In order to achieve a good accuracy, the CNN has to be trained with a set of images (dataset). The process of training the CNN can be described in a simplified manner with the following steps:

- Run the CNN with with the dataset
- Modify the weights
- Evaluate the accuracy

For each iteration on the training data the weights gets more correct and the models ability to accurately classify a picture better. The size and content of the dataset used to train the model will also impacts it accuracy. Typically, the available dataset is divided into three subsets; training, validation and test. The training dataset contains the images used for training, the validation dataset is used to evaluate the model accuracy during training, and the test dataset is used at the end to verify that the model has generalized a pattern as opposed to having memorized the training images (overfitting).

Training a neural network from scratch is computational intensive in itself, and more so because it can be necessary to experiment with different network designs. A number of user friendly wrappers for CNN frameworks and implementations of pre-defined models can be found online. For example at https://github.com/tensorflow/models.

DIABETIC RETINOPATHY DETECTION

In 2015 there was a competition to develop an automated method of detecting a type damage to the eye caused by diabetes known as diabetic retinopathy. The competitors would use color fundus photography as input and use image classification, pattern recognition, and machine learning to "push an automated detection system to the limit of what is possible" – ideally resulting in models with realistic clinical potential.

The contestants had over 35,000 retina-images available for training, and there were in total 5 severity classes. The distribution of classes was fairly imbalanced and most of the images had no indications of the disease. A few percent had the two most severe ratings.

Min-Pooling - The winner of the competition, used a convolutional network running on an NVIDIA Graphic Processing Unit (GPU). Images were pre-processed using Python and Open Source Computer Vision (OpenCV). Preprocessing included scaling to achieve uniform sizes, subtracting color information, and cropping the image. In addition - images were scaled and rotated to achieve a larger training data set. The Network consisted of 12 Layers, each with between 5 and 356 connections.

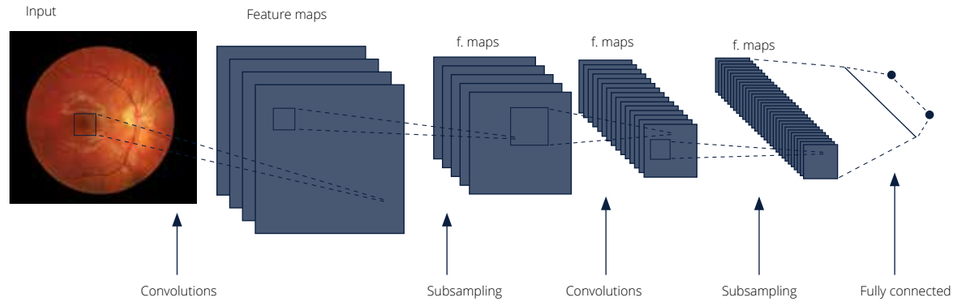
The achieved score of the neural networks was measured as a quadratic weighted Kappa score, and the winner achieved a score of 0.85. Research measuring how well an ophthalmologist can detect diabetic retinopathy find values ranging from 0.72 to 0.838 kappa.

CNN FRAMEWORKS

During the AMD diagnosis research performed by Data Respons and HSN, two popular CNN frameworks were tried out. Tensorflow is Google's framework for building neural networks that was released as open source in 2015. Several tutorials for designing and training CNN's using Tensorflow are available online, and also more user friendly wrappers like TF-Slim. Theano is another open source numerical computation library, that is primarily developed by a machine learning group at the Univerité de Montréal. Other Popular Frameworks include Caffe, Torch and Deeplearning-4Java.

Theano is widely used for deep learning research in academia. Raw Theano can be somewhat non-intuitive to use, and for this reason several wrappers for working with ANN's have been developed, including the open source libraries Keras, Lasagne, and Blocks. Some of the creators of Theano later went on to create Tensorflow at google and the two libraries have some similarities.

Theano and Tensorflow both let the user define computational graphs in the Python language, which is a powerful, high level scripting language. The computational graphs consists of nodes that can be functions, variables or constants, and enables the definition of computations involving multidimensional arrays. Convolutional neural network models map very well to computational graphs. After a network model has been defined as a computational graph, the framework compiles it to executable code. The model can then be trained and used without the speed-penalties of using a scripting language. Of Theano and Tensorflow, Tensorflow has the most feature-rich toolset for visualizing the network models.



HARDWARE

Training a CNN for good accuracy may require thousand of iterations on data sets containing tens of thousands of images. There are ways to optimize the input data for speed, such as reducing image resolution, removing color channels and pooling. Even with these measures, the architecture of normal Central Processing Units (CPU's) are not very well suited for training neural networks. Normal CPU's process data using only a small number of cores, but training models is a task suited for more massive parallellization. Modern High-end GPU-s can have thousands of cores, and several cards can be incorporated into a training rig. For this reason, many machine learning frameworks have support for GPU computing.

> The tools for building image classifiers using neural network has become readily available the last years, and can perform better than humans in some circumstances.

There are two common low-level interfaces that a CNN framework can leverage for offloading computing tasks to a GPU. All modern Nvidia GPU's implements the Compute Unified Device Architecture (CUDA) platform and API for parallell computing. AMD was an early adopter of the Open Computing Language (OpenCL). OpenCL is an open platform for heterogeneous computing, and is supported on both AMD and Nvidia GPU's. If considering buying hardware for training convolutional networks, the choice in hardware depends on the CNN framework that will run on it. From the CNN frameworks described above, Tensorflow supports CUDA and Theano supports both CUDA and OpenCL.

SUMMARY

The tools for building image classifiers using neural network has become readily available the last years, and can perform better than humans in some circumstances. Already in 2015 both Microsoft and google's image classifiers beat the human score in the ImageNet challenge (http://image-net.org). Software frameworks and learning material for building neural networks are available online and for free, but can be challenging to navigate. For initial experimentation and benchmarking - it can be smart to train prebuilt models. A number of models and wrappers can be found for the popular frameworks Theano And Tensorflow.

INSIDE WRITERS

HALDOR HUSBY

Principal Development Engineer
Data Respons

MASc Electrical Engineering, University of Toronto
Siv.Ing. Electronics, Norwegian University of Science and Technology

FREDRIK BAKKE

Senior Development Engineer
Data Respons

MSc degree in Systems Engineering,
Buskerud and Vestfold University College

SVEIN TORE EKRE

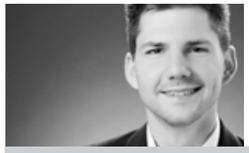
Senior Development Engineer
Data Respons

MSc degree in Systems Engineering,
Buskerud University College

JON-HÅKON BØE RØLI

Development Engineer
Data Respons

MSc in Engineering Cybernetics,
Norwegian University of Science and Technology



THIMO KOENIG

Senior Software Developer
Data Respons, GmbH

M.Sc. Business Process Engineering,
FHDW - University of applied science

CHRISTINE MITTERBAUER

Senior Software Engineer, Project Lead
Data Respons, GmbH

Dipl.Ing. Technische Informatik, DHBW Stuttgart

EIMUND STRØM

Specialist Development Engineer
Data Respons

Bachelor of Science, Computer Science
Buskerud University College

LARS ALBERT FLEISCHER

Senior Development Engineer
Data Respons

Cand. Scient, University of Oslo.

PUBLISHER:
Data Respons ASA,
Sandviksveien 26, 1363 Høvik
Tel: +47 66 11 20 00
info@datarespons.no

EDITOR-IN-CHIEF:
Kenneth Ragnvaldsen
CEO, Data Respons

EDITOR:
Elisabeth Andenæs,
Corporate Communications & Brand Manager, Data Respons
Tel: +47 92 20 30 03
Email: ean@datarespons.no

TECHNICAL EDITOR:
Ivar A. Melhuus Sehm
Director R&D Services, Data Respons
ise@datarespons.no

Norway
Oslo, Høvik, Kongsberg, Stavanger, Bergen.

Sweden
Stockholm, Gothenburg, Linköping

Denmark
Copenhagen, Århus

Germany
Berlin, Munich, Stuttgart, Erlangen, Karlsruhe

Taiwan
Taipei

Main offices

Group HQ
Data Respons ASA
Sandviksveien 26
NO-1363 Høvik, Norway
Tel.: +47 67 11 20 00
info@datarespons.com

Germany
Data Respons GmbH
Amalienbadstr. 41, Bau 53
DE-76227 Karlsruhe
Tel.: +49 721 480 887 10
info@datarespons.de

Sweden
Data Respons AB
Jan Stenbecks Torg 17, III
SE-164 40 Kista
Tel.: +46 8 501 688 00
info@datarespons.se

Denmark
Data Respons A/S
Smedeholm 10
DK-2730 Herlev
Tel.: +45 88 32 75 00
info@datarespons.dk

Norway
Data Respons Norge AS
Sandviksveien 26
NO-1363 Høvik
Tel.: +47 67 11 20 00
info@datarespons.no

Taiwan
Data Respons ASIA
18F-6 NO. 738,
Chung-Cheng Road,
Chung-Ho, New Taipei
Tel.: +886 2 8226 2150

Sylog AB
Jan Stenbecks Torg 17, III
SE-164 40 Kista
Tel.: +46 (0)8 750 49 00

TechPeople A/S
Smedeholm 10
DK-2730 Herlev
Tel.: +45 88 32 75 00

MicroDoc Computersysteme GmbH
Elektrastrasse 6A
D-81925 Munich, Germany
Tel: +49-89-551969-0

Our values
Being Generous
Responsibility
To Perform
Having fun

Digitalisation
of the industries of tomorrow

We can develop everything from sensor level to the mobile app, making us a good partner for our customers with their digital transition.

data respons
EMBEDDED SOLUTIONS