WHITE PAPER

Intelligent Flexible IoT Nodes

Exciting new opportunities and deployment models. Heterogeneous system on-a-chip (SoC) architecture. Benefits of distributing intelligence across all nodes. Best practices to address next-generation nodes for the Internet of Things. Smart and always aware.

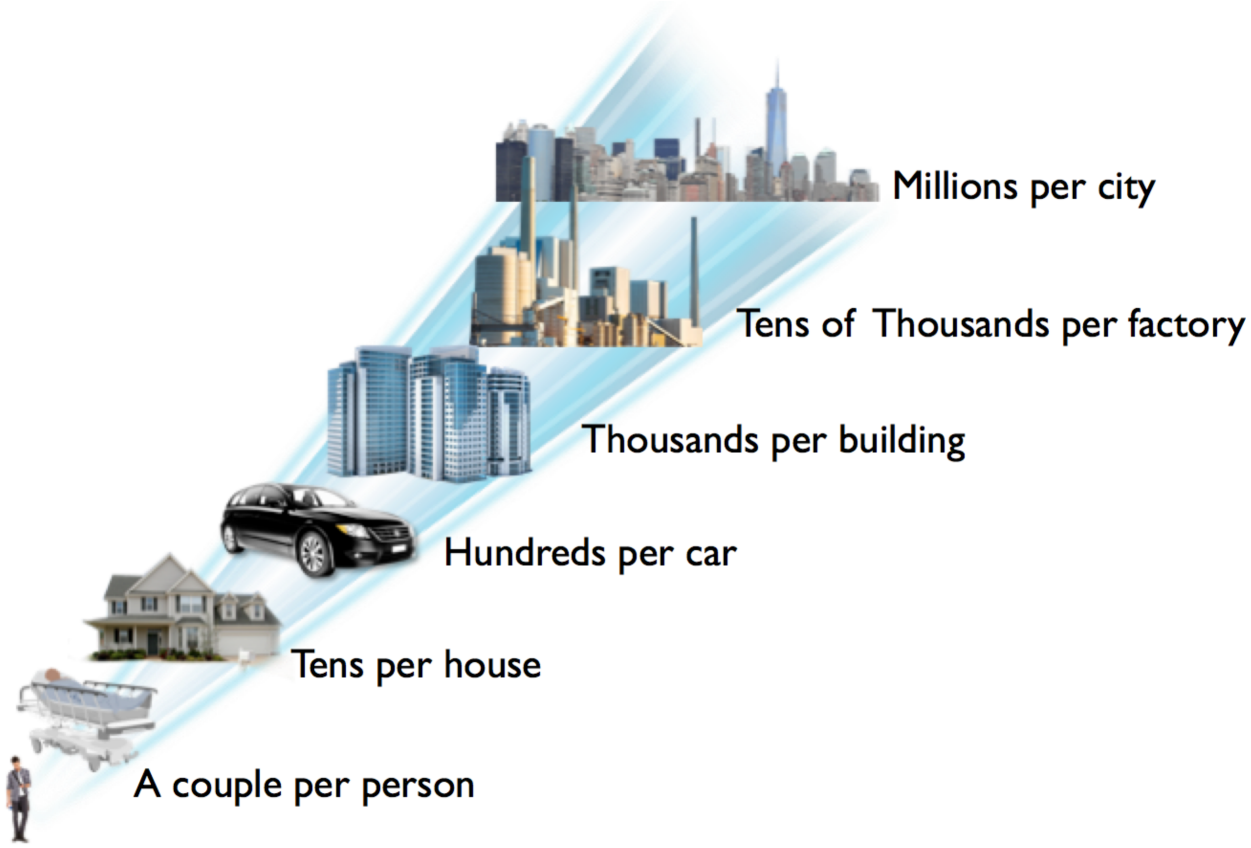
ABSTRACT

The media is buzzing with articles every week about the benefits of deploying IoT systems across various industries. Some forecasts mention the trillions of dollars that will be saved on a global scale from such deployments and the dramatic boost to those companies that make it all possible. At the heart of this global deployment is the IoT node, a sensor that converts the physical world to digital information. Most forecasts mention 50 billion nodes and beyond to be deployed by 2020. The challenge for companies wanting a part of the market is how to address the thousands of applications with the minimum number of devices. Clearly one node is not going to address all markets in an economically viable manner. The goal is then to find a common architecture from which to build a minimum set of nodes to address a maximum number of markets. This paper proposes an elegant architecture to address all nodes for all markets.

The road to 50 billion nodes

IoT systems are exciting and they are set to become even more powerful and pervasive. These systems are being deployed everywhere as they can realise big gains in operating efficiency, with 20% to 30% a commonly quoted figure. More important, IoT systems open the door for new revenue streams from data services.

Usually, it is good to have an idea about the size of a market before investing in products for that market. IoT is a special case as it represents a horizontal trend or deployment model across many different market segments. Below is a proposal for a method where each person may easily create his or her own precise model.



Consider the market size based on where the IoT node will be located. If it is something that humans will wear, then a few billion humans each wearing a few nodes will realise a total of tens of billions of nodes. For cars, there are approximately one billion cars in circulation today with 80m to 90m more being shipped every year. A few nodes per car add up to a few billion units. More people are moving to urban areas at a faster rate. By 2020 a few hundred cities will each have over one million inhabitants. Consider all the objects in each city such as lights, garbage bins, traffic lights, news kiosks, mailboxes, etc. that will be fitted with IoT sensors. Billions of nodes will be deployed at a global scale. The market numbers for each of these segments are easily available online from many sources. The segments above are not exhaustive, there are many more such as agriculture and mining that are also deploying IoT systems.

Connected, always on and always aware

The function of all these nodes is to be always aware and report changes in context. Different segments have different requirements as to how often to sample the physical environment or context and when to report upstream. This seems to be a straightforward statement but it hides two key issues. The node is required to be always on to be always aware and it needs to report changes.

No one would reasonably propose to deploy 50 billion nodes that are wired to the network and connected to a power source. Economically, this would not be a viable solution, as the cost of deployment will totally outweigh any potential benefits. The logical deployment model calls for mainly battery operated, wirelessly connected nodes. Changing batteries on a few billion nodes every few months is also not an option, however. Hence, battery life is required to extend over multiple years. In the case where end points do have a power source, they still need to be highly energy efficient since they need to be always on. A few billion nodes that are always on with sub-optimal power consumption would add a huge burden to the electric grid.

Connectivity opens the door to many new business opportunities. Yet again, one has to carefully design around upstream reporting with radio transmissions, both for energy efficiency reasons and to take care so as not to flood the upstream network with useless data. A few billion nodes sending streams of useless data upstream would introduce unnecessary stress and cost to any infrastructure.

To design a node

In essence, what is needed besides the best low-power technology is a node that operates in a smart way to maximally reduce power consumption. Each node must have the minimum viable level of intelligence based on the targeted segment in order to address these two orthogonal needs of always aware and low power consumption.

IoT Nodes may be classified by performance or by functionality. Using a performance-based classification is good when looking at the components required when building the node while a functional approach is focused more on the system design from end to end in terms of how the intelligence is distributed.

Performance based classification:

1. Ultra-constrained nodes. Using an RTOS or bare metal with 16K of RAM. Such a node may be doing energy harvesting and would limit radio transmissions to the minimum required, to conserve power.

2. Constrained nodes. Using an RTOS with 32K-64K of RAM. Most likely running on a battery where all the software is optimised for battery life. Again limiting radio transmissions to a minimum.

3. Mainstream node. Using a feature rich RTOS with 128K of RAM such as ARM® mbed™ OS. The context gathering operation starts to get more complex since there is room to do more local operation, as opposed to sending all data upstream.

4. Gateway node. Using an advanced OS with 64MB of RAM. This is a sophisticated node that can run advanced software and complex algorithms and is mostly running from mains power. This node will have multiple radios to support the local network.

Functional based classification:

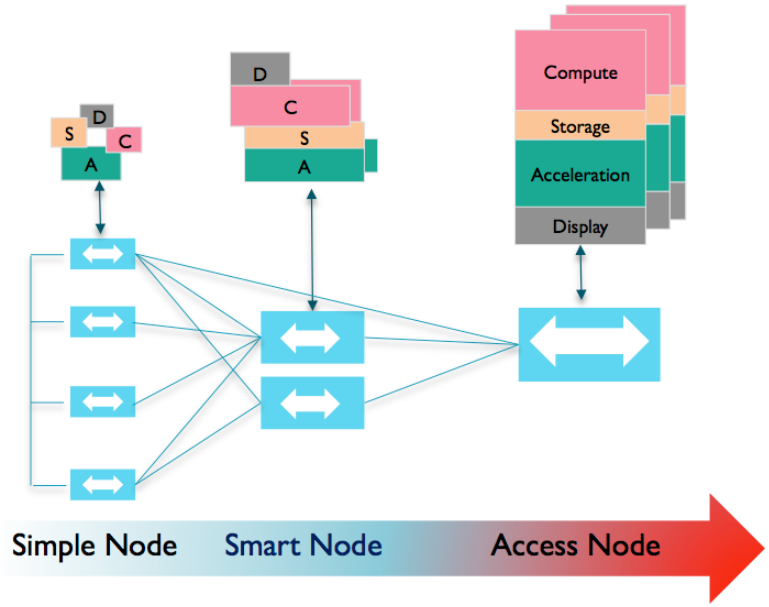
1. Simple node. A node that is not aware of the rest of the local network. The functionality is to collect and report context information to the specified destination.
   Any of items 1 to 3 above may be used for simple nodes.

2. Smart node. A node that is fully aware of all other nodes in the network mainly via sophisticated software that understands mesh networks, local topologies and authorised interactions between nodes in the same network.

3. Access node. This is the box at the edge that connects the local network to the Internet via whatever broadband link appropriate for the application. The access node will have multiple radios facing the local network.

There are no restrictions on creating a node that is both a smart node and a gateway. The classification above makes the separation only to highlight the different functions in the local network.

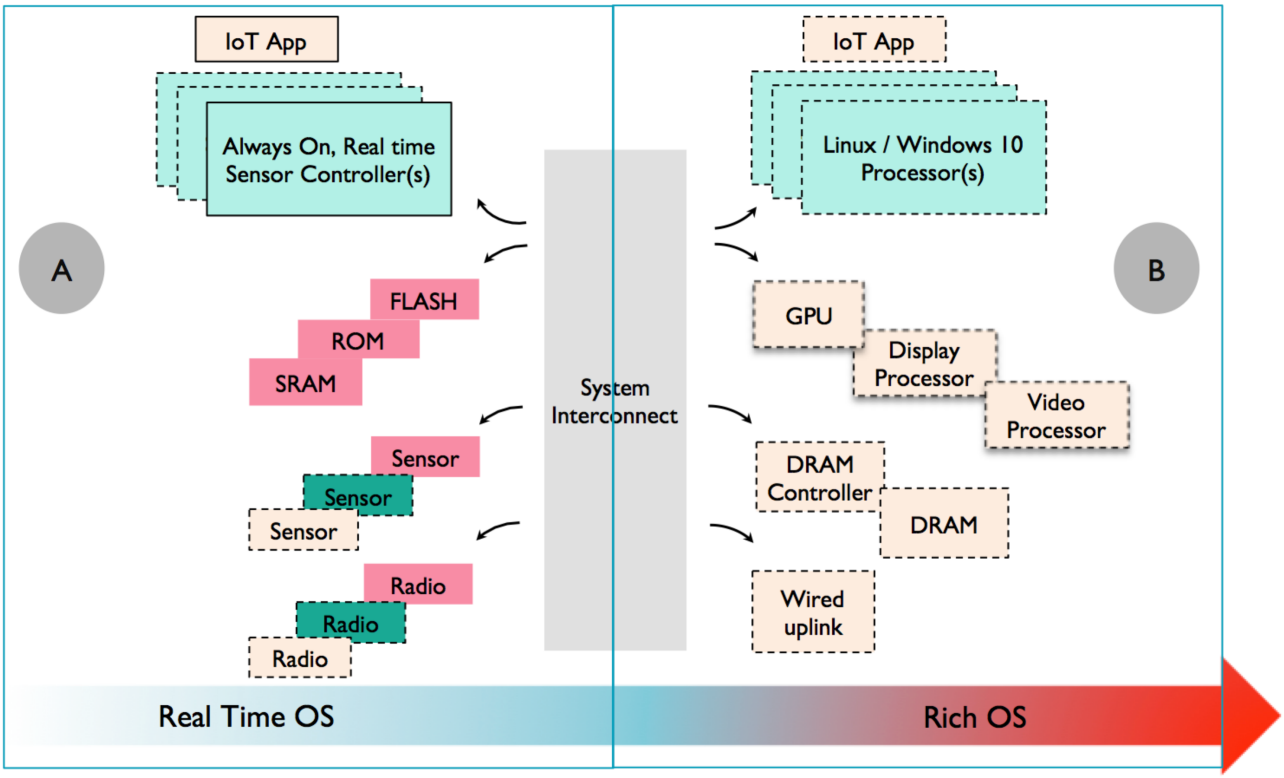
The different requirements for all the classes above are met with the right dose of intelligence and features for each node. We can apply exactly the same clever scheme to all IoT nodes. The software platform, features and deployment segment dictate the components included to build a specific class of nodes.



Under the hood

Inside every node there has to be an always on, always aware part to which we bolt on the standard processor system required to run a commonly used feature rich operating system. Here is a detailed explanation of the components and how they fit the requirements. In figure 1 below, the A side of the diagram represents the design of IoT nodes that use a real time operating system to host the IoT application. We have the following components:

Figure 1 - Architecture of an IoT node



Always on sensor controller(s)

For software productivity, we need a 32bit real time embedded processor from the ARM® Cortex®-M product line. In IoT there is a need to quickly develop and release a large amount of software that will handle connectivity, security and IoT applications while properly managing and controlling one or multiple sensors that may have real time deterministic requirements. Cortex-M processors were designed specifically for these requirements. Software productivity is important since one node design is likely to be used to release products for multiple different markets.

In some cases, multiple Cortex-M processors maybe required, one to handle radio stacks and connectivity, one to manage the sensor and a third to run the whole system and network membership. It is very easy to use multiple processors of the same or different model since the same bus is used connect to the system fabric. Multiple processors in many cases are better for power consumption since only the right amount of processing power is active at any moment in time to handle the task of the moment. For example, there is

no need to wake up the main processor if there is no data transmission required but the radio processor may need to be checking for incoming data all the time.

An architecture using three processors will almost certainly be a good fit for 99% of applications. In many cases, one core is more than enough. From a design perspective, two products would be economically viable to service the whole market but one has to always overlay the economic viability for each segment.

Memory subsystem

Most real time embedded designs use flash memory to store the program, SRAM to store code and data and ROM to hold the basic system description. The size of each of those memory blocks depends on the system configuration and the intended operation and software complexity.

It is not that difficult to have one design service many segments since one memory size fits many applications and having a bit of extra space is always good for future expansion. It is only when the requirements of an application call for a substantial increase in memory that one needs to consider creating distinct products since there is no need to burden the smaller one with the cost of the extra memory that will definitely remain unused.

Sensors

IoT is about converting the physical world into digital information. Without sensors there would be no IoT. This is the component that drives the expansion in the number of distinct products required. Different market segments require a different set of sensors. This is why many companies today choose to integrate many interfaces as opposed to the sensors. An interface may be common to ten sensors, so instead of making ten products one can serve all ten markets with just one device. Eventually the sensors may need to be integrated for cost reasons but that could be done as the application becomes clearly identified and deployment ramps.

Radio

At this time, there are quite a few radio standards in IoT: Bluetooth® LE, Wifi, 802.15.4, RFID, 802.11ah, and LTE Cat-0. Then we have a few more long-range, low bit rate, and other proprietary standards that compete for market share. There will not be one radio standard that dominates or covers all deployments. There will always be many different standards that need to be supported. Hence, like sensors, for now integrating an interface to the radio is sufficient but eventually the radio has to be integrated as mass deployment reaches a tipping point.

IoT App

On top of the hardware, there is the application that performs the IoT function. This is the piece of software that will link the sensor data to the network. Here we have unlimited room for innovation. People have only just started in this space and there is no limit to the sort of deployments and IoT mash-ups that will be deployed in the future.

Simple Node Summary

The design of part A in figure 1 properly addresses a large number of segments where the IoT application is hosted using a real time operating system. A variety of Cortex-M processor tool chains enable sophisticated yet easy 32bit IoT application development while satisfying the need for low power and determinism. It is interesting to guess on the variety of parts that may be produced given this design. If we make the following approximate calculation:

3 processor group designs x 5 memory size configurations x 20 sensor combinations x 10 radios

Then we get 3000 different parts to cover all markets. Room for innovation and differentiation, all based on a simple design that meets all requirements for IoT.

On the other hand, there are many reasons why an IoT application may need to be hosted on top of a rich OS. For that case we would add part B to part A as shown in figure 1, thus creating an extended solution that maintains the real time deterministic low power always-aware section and adds in a rich operating system to host the IoT applications. Other compute or acceleration blocks are also added as required. The one thing to keep in mind is that an operating system such as Linux or Windows 10 requires a sophisticated arrangement for system fabric and memory, unlike the case for real time operating systems. One of the key factors in power consumption under such an OS is the DRAM memory subsystem. The Cortex-A7 processor utilizes many different techniques to maintain high efficiency, but there is not much it can do about the power consumption of DRAM.

At this point it is natural to wonder about the logic behind keeping the processor in section A above when we have a processer in section B. The answer is straightforward. A processor that runs a rich OS requires a memory management unit (MMU), which then removes any possibility of determinism in handling of sensors. This is the main reason why most embedded designs use a real time OS and no MMU. The other reason is that such an OS requires large memory. Current memory technology uses DRAM that is just too power hungry for always aware applications. Hence, we keep the Cortex-M in section A for always aware operations and we use the Cortex-A in section B for the occasional IoT application intervention.

Rich OS processors

It is very common in the market to use Cortex-A7 to host a rich OS. The Cortex-A7 is a high efficiency 32bit processor that can be designed in uniprocessor or multi-processor arrangements with caches. For example, a few hundred million smart phones have already shipped with quad core Cortex-A7 processors running the Linux kernel. Boards with such quad core devices are available on the market for a few dollars and are a perfect fit for section B above for IoT access nodes. They just need to be augmented with the low power always-aware part.

Many companies prefer to use such an OS due to the availability of sophisticated software frameworks to build on. This again highlights that software productivity is a key factor for IoT deployments. As a final note, other models of Cortex-A processors may be used depending on the complexity of the node.

GPU, Display and Video processors

The IoT application may require a display, such as a thermostat, or may require some acceleration for video capture or other such operations. These additional compute blocks or accelerators are added to address specific segments. It is a delicate balance from a power consumption perspective to when it is more efficient to do an operation in software versus doing it in a dedicated compute block. This is an area for innovation as deployment of IoT systems spreads across all segments. In many cases, an accelerator block will make it economically viable to deploy in areas that were not accessible before with pure software solutions.

DRAM controller and DRAM memory

A rich OS requires large amounts of memory, which in these days uses DRAM technology. This memory requires a controller and both of these parts increase the power consumption of the system. This is a key reason why part A in figure 1 is required in the system since a DRAM based memory system can not possibly be always on in order to perform the always-aware functionality.

The size of the required memory is normally well defined in advance based on the Rich OS to be used.

Wired uplink

Access nodes may have a wired or a wireless uplink to the network infrastructure. We have fiber, Ethernet, cable and a few others. These are very established technologies these days with very low deployment risk.

IoT App

This is the same as before except that the app is hosted on the processor in section B instead of section A to utilize the sophisticated software frameworks.

Complex Node Summary

Given all the above we have:

- Uniprocessor, two cores, four cores.
- Multiple Cortex-A processor models
- Small, medium and large memory configurations. At least 20 accelerator configurations come to mind.

9 processor group designs x 3 memory size configurations x 20 accelerator combinations x 5 broadband options:

This yields 2700 different parts to cover all markets. That would bolt on next to the 3000 different parts above. This is nearly 9 million different product configurations to build IoT with over the next few years. This is a wide open space for innovation and differentiation, all based on a simple design that meets all requirements for IoT.

The opportunity ahead

Why should any one spend any time to study this architecture? The answer is tightly connected to the promise and opportunity presented by IoT. If it is true that every object known to mankind is going to be connected to the network then the size of the opportunity is beyond all forecasts. The market is too large to let it pass by without making an attempt to capture a share. The proposed architecture reduces the risk and the investment. One architecture, one design, many variations to fit many product markets.

The other advantage is that all the experience gained and success achieved in one segment will be immediately applicable to many other segments. Change a sensor here, swap in a new radio, add a new IoT App and the result is another different product for a different segment. A small fractional investment yields a large gain. The situation gets even better if a position can be taken in the data services and analytics piece of the market. Why stop at offering only IoT nodes when one could also offer the associated data services for those nodes? After all, big data starts with little data.

Summary

In this paper we have presented the fundamental two requirements for all IoT nodes; being always aware and maintaining the maximum possible energy efficiency. We then introduced an architecture that meets those requirements across many market segments. This architecture is suitable for the many different designs that IoT segments demand. The investment and learning to produce the first design map directly to a wide range of markets with subsequent incremental variations.

The design is based on two main sections, one that is always present and the other being optional based on the target application. The required section relies on using low power 32bit processors from the Cortex-M family to handle the always-aware real time deterministic functionality. The optional part adds 32bit/64bit processors from the Cortex-A family to carry the rich OS. Using both sections leverages the best of two worlds to satisfy the requirements and to produce the wonderful variety of products that IoT will be built from.

Future white papers will focus on the security aspects, software design and ecosystem for IoT nodes.

If you are looking to design such a node then we would encourage you to contact ARM product management to discuss in detail how, together, we can enable you to build the most successful solutions.

FOR MORE INFORMATION

For more information on the intelligent flexible IoT nodes and ARM processor design, visit http://www.arm.com/

Authors' biography

Kinjal Dave is a CPU Product Manager at ARM. He is responsible for managing the high efficiency range of Cortex-A class processor products, such as Cortex-A5, Cortex-A7 and future roadmap processors. He joined ARM in 2007and worked on hardening ARM CPUs in the processor implementation team before taking on a product management role of ARM CPUs in 2012. Prior to ARM, he worked with Cadence, India as a Product Engineer. Kinjal holds a Masters degree in Information & Communication Technology from DAIICT, Gandhinagar. He holds 1 patent.

Diya Soubra is a Senior Product Marketing Manager for Cortex-M processors at ARM. He has 20 years of experience in the semiconductor industry, during which he held various positions in engineering, product marketing and business management. Just prior to joining ARM, he worked with hi tech start-ups to develop their business. Before that he was in charge of product marketing for devices for VoIP and broadband gateways. He has also developed various software and hardware products for communication protocols and infrastructure systems while working for Rockwell Semiconductor, Conexant Systems and then Mindspeed Technologies. He received a B.S. in Electrical Engineering from the University of Nebraska at Lincoln, a Masters of Science in Engineering from the University of Texas at Austin and his MBA from the Edinburgh Business School. He holds 1 patent.