Adding Intelligence to the Next Generation of Smart Devices

By Kristopher Ardis, Executive Director, Micros, Security & Software Business Unit

May 2018

maxim integrated™

Abstract

As internet of things (IoT) applications become more mature, we see devices wanting to sense and process more data, but we also see limitations being exerted on communications: trying to filter data to minimize the bandwidth required, or even trying to push decisions from the cloud to the local device.

Introduction

Making more intelligent devices ensures a positive ROI from our smart things

Things have intelligence

I'm not suggesting that inanimate objects have brains. But in my white paper, "Powering the Next Generation of Smart Devices," I wrote about what your house's door could tell you if it could speak, and what it could do if it could listen to you. Your door with a brain would be pretty useful to you, but perhaps not to your neighbor or someone living on the other side of town. Let's think more broadly: what if your street light had a brain? We already see some examples of smart street lighting that can sense when there's no traffic and dim lights to save money or report light outages quickly to improve safety. But street lights represent valuable real estate being deployed across an entire city or an entire highway…in fact, there are very few things with such a valuable footprint. More intelligent street lights could yield more accurate weather maps or pollution data across the city. They could listen for car wrecks or gunshots to alert the appropriate first responders. In this case extracting more intelligence out of our devices does something for society, not just for the person who owns the device.

The things we have around us and interact with have intelligence. That's why we want to connect them to the Internet. But many of these things don't have a way to share their knowledge or listen to our requests—the technology hasn't been there to make it worthwhile or practical to connect everything that might have valuable data.

We've already talked about power being a challenge to unlock this intelligence, but so is the act of adding the ability to process data (or 'adding intelligence'): how much processing capability needs to be added to a device? How do you add processing capability without overdoing it (and going over your cost budget) or under-doing it (and delivering a product whose 'intelligence' is in question)? How do you make sure your device continues to deliver a positive return on investment (ROI) when you may have to deal with changing requirements once the device is installed? What if in two years all the street lights in your city want to also listen for screams, in addition to gunshots and car crashes? Will you have enough processing power? Will you have enough memory to add features?

We are surrounded by an invisible intelligence, waiting to be freed. But installing devices to free that intelligence is risky – there is always an ROI calculation, so you may only have one chance to deploy street lights in your city. How can we maximize ROI when we may not be entirely sure about everything we want our device to "return" to us?

Figure 1. A digital thermostat is one of the many devices in our homes today with invisible intelligence.

What is Invisible Intelligence?

In the white paper "Powering the Next Generation of Smart Devices" we introduced the concept of 'invisible intelligence.' In brief, 'invisible intelligence' rests on three principles:

1. The things we interact with and have relationships with have data, and when enabled by technology, this shouldn't fundamentally change the way we interact with them. We still set the dial on a smart thermostat, but the thermostat starts to learn our preferences based on a number of conditions. We still glance at our wrist to tell the time, but we are presented with other meaningful data that we can explore further if we wish.
2. We shouldn't notice that the smart device is fundamentally different from its 'dumb' counterpart, thus it is invisible. A window sensor that is so large it blocks part of the light from the outside defeats the purpose of the window.
3. The data from our smart device needs to have value. Invisibly intelligent things will inherently be more expensive than their dumb counterparts. A smart brick might be easy to use and won't change how you interact with or think about bricks, but is it really a good return to buy more expensive bricks that will send you a text message when they fall off your house?

Let's consider the three principles of invisible intelligence and how they pertain to our original question: how do we make sure our IoT devices have the right amount of intelligence?

- Our interaction: our smart devices may need to be updated over time, but as users we generally won't want to know or have to take action to make sure our smart irrigation system can continue to get weather forecasting data. If as users we need to assist in performing a firmware update on a device, our patience with the device is likely to expire before the useful life of the electronics. On the other hand, if our smart irrigation system can't accept a seamless firmware upload (and adopt new features) because it runs on an older processor or doesn't have enough memory, we aren't likely to buy the next-generation device. Cell phones may have us all trapped in the

game of upgrading every couple of years, but it isn't likely consumers will have the patience to do that with the rest of their smart devices.

- The device's appearance: There is a danger in making the device appear to have too much intelligence and in making the device appear to have zero intelligence. Simple irrigation controllers are complex enough to program, but if the touchscreen on a smart irrigation controller similarly shows dozens of cryptic configuration options, it would be too intimidating for users to adopt. Likewise, if the smart irrigation controller features mechanical switches and buttons, it wouldn't feel 'smart enough' and likely wouldn't make the trip home from the store.
- The data's value: In a smart irrigation system, the value is in the device's ability to combine your desired schedule with weather forecasts or other external factors to make smarter decisions about watering plants. You might imagine that a smart irrigation controller may want to communicate with as-yet-uninvented sensors in the future, so designers would need to plan for excess processing horsepower and memory to accommodate for added features. But too much excess processing horsepower and memory drives up the cost of the device, lowering the value provided by the intelligent controller's ability to analyze data and make smarter decisions.

Added Intelligence in the Wild

We already see examples of devices that do a decent job adding 'invisible' intelligence to traditionally non-smart devices. The Nest learning thermostat is a good example--our interaction with a Nest learning thermostat seems very much like our interaction with traditional thermostats where we use the dial to set the desired environmental condition. For appearance, the Nest thermostat clearly looks modern and advanced compared to most traditional thermostats, but it is somewhat obviously still a thermostat. And there appears to be value in the data it generates and the savings it provides.

We can see examples from the industrial sector, too. Capstone Metering's IntelliH2O® is a water meter that seeks to go far beyond measuring consumption and billing, and it does so while exhibiting the characteristics of invisible intelligence. Our interaction as consumers with a water meter is already fairly invisible—we use water, and we receive a bill in the mail for our usage. For the utilities who purchase, install, and use the IntelliH2O, their interaction is certainly more complex, but similar to that with other communicating smart water meters. Even the replacement cycle for the IntelliH2O is comparable, if not better, than that of other communicating water meters: with a rechargeable energy-harvesting battery, it should outlast other smart water meters and even approach the useful lifetime of traditional mechanical meters. From an appearance standpoint, Capstone Metering has an easy job here:

water meters just need to fit in about the same amount of space as the mechanical meters that are being replaced. But where IntelliH2O really shines is in the third aspect of invisible intelligence: that the data should have value. IntelliH2O can exist in our homes and we are never the wiser, but it can also alert us via text message that we have a toilet leak, or that there is some other suspicious water consumption in the house before it is too late. It can also help utilities identify water pressure anomalies—high water consumption (and billing) is often associated with high water pressure in the system while residents are watering their yard, which is almost always done on time (water each zone for five minutes) rather than volume (water each zone with 50 gallons of water). In addition, pressure monitoring can help identify risks for pipe bursts prior to them happening, saving utilities millions in emergency repairs. "In building the IntelliH2O platform, we took the philosophy that more sensors, more intelligence, and the ability for the technology to become a force multiplier would provide our customers and society with the best solution for responsible management of water," says Scott Williamson, CEO of Capstone.

There are plenty of examples of devices becoming more intelligent while adhering to the principles of 'invisible intelligence', but the universe of IoT devices is still in its infancy and there remains a lot of information to liberate. What are the considerations that these 'dumb' devices should take in order to join the 21st century?

Capstone's smart water meters deliver valuable Insight

Figure 2. Capstone's IntelliH2O water meter.

The Basics: Running Algorithms, Network Stacks, Operating Systems, and Whatever Else

Your existing device may not have any electronics at all or it may be exceedingly simple. Simple smoke detectors and window sensors today use 8-bit microcontrollers with only a few kilobytes of code memory to perform their jobs. But once you start talking about a smart, communicating device that may need to monitor multiple sensors and upgrade itself in the field, that 8-bit microcontroller will quickly become overwhelmed handling those requirements:

- Running multiple algorithms: Algorithms generally mean the processor has to do some heavy lifting such as: signal processing, lots of computations, and lots of intermediate storage. This means 8-bit micros simply have to execute more instructions and burn more power to do a single math operation that a 32-bit microcontroller can execute in a single instruction cycle.

  For an example, here's a 32-bit add on an 8-bit microcontroller and on a 32-bit microcontroller. As you can see, one of these is more efficient than the other!

```
ADD32ON8BITMACHINE:
   anl PSW, #0E7H;Register Bank 0
   mov a, r0 ;load X low byte into acc
   add a, r4 ;add Y low byte
   mov r0, a ;save result
   mov a, rl ;load X next byte into acc
   addc a, r5 ;add Y next byte with carry
   mov rl, a ;save result
   mov a, r2 ;load X next byte into acc
   addc a, r6 ;add Y next byte
   mov r2, a ;save result
   mov a, r3 ;load X high byte into acc
   addc a, r7 ;add Y high byte with carry
   mov r3, a
   mov C, OV
   ret

ADD32ON32BITMACHINE:
   mov a, r0
   add rl
   mov r0, a
   ret
```

- Microcontrollers with DSP or floating-point acceleration will be able to execute many algorithms faster than those without: In general, throwing hardware at a problem means less energy consumption. While the hardware might instantaneously consume a little more power, it will run for a significantly shorter period of time than software doing the same thing.
- Managing a network stack: Smart devices often mean communicating devices. Network stacks require code and data memories to operate. And depending on the network, the memory space could be considerable. Imagine if a device didn't have the memory available to remember routing tables (or forgot its routing table every time it went to sleep). Then every piece of data it needed to transmit would be accompanied by a long period of completing address resolutions...wasting power and network bandwidth because the device didn't have the memory capability to remember what its relatively static network looked like.
- Using an operating system: As devices become more complex (more sensors, more applications, more network capabilities), it makes sense to use a real operating system to manage core resources like processor bandwidth and memory allocation. But even the operating system itself needs some memory and computing resources to function.
- Securing data and commands: If you're not thinking about security for your connected device, you're already behind. Innocuous devices like connected cameras have been used to take down major web services like Netflix. More serious infrastructure like electric utilities must be even more guarded. The algorithms used to secure data and commands are complex and highly math-intensive—processors need enough memory space to store the algorithms, working memory for intermediate results, and processing horsepower to compute ciphers in an amount of time that doesn't interfere with the application's need for reactivity. Here's another case where 32 bits is better than 8 bits, and where specialized hardware can be even better for saving power and time compared to software-only implementations.

Sizing Up the Market

There is a universe of processing options available...so many that it may become daunting to select an option. Before digging into some of the parameters critical for building invisibly intelligent devices, we can broadly look at the types of processors available for designers:

- Microprocessors: This class of product includes processors with typically 32-bit or wider datapaths, that usually run at very high performance...hundreds of megahertz or even gigahertz. Microprocessors also usually have true memory management hardware, allowing them to run operating systems like Linux. Microprocessors rely on external memories, and because they generally run large

The smarter the device, the more it needs for processing, network stacks, and security

operating systems, they need many megabytes of code and data space. Their power consumption is measured in hundreds of milliwatts up to several watts. These products are appropriate for computationally intensive or user interface-heavy applications; however, their power consumption requires connection to a constant power source or regular recharges. Your phones and tablets use microprocessors, and need a recharge after several hours of use. Processors in this class include MIPS, Arm® Cortex®-A, and Intel cores.

- 8-bit and 16-bit microcontrollers: On the other end of the spectrum are 8-bit and 16-bit microcontrollers. With integrated code memories as small as a couple kilobytes and as large as a couple hundred kilobytes, these microcontrollers are more suited for smaller applications where inputs and outputs are limited. Processing speed is usually limited to a few megahertz, and power consumption can be much lower with this class of products, measuring a few milliwatts. Processors in this class include architectures such as 8051, MSP430, and AVR.
- 32-bit microcontrollers: In the middle are 32-bit microcontrollers such as the Arm Cortex M-class of products. These processors typically have code and data memories integrated on-chip, and code space can be as small as a few kilobytes and up to a few megabytes. Occasionally, these microcontrollers may include options to access external memories as code or data. The 32-bit microcontrollers typically run from a few megahertz to a couple hundred megahertz. They generally do not have a complete memory management unit, so operating systems like Linux cannot be used, but there are a variety of smaller and real-time operating systems suitable for these products. Power consumption for 32-bit microcontrollers can be as low as a few milliwatts, comparable to 8-bit and 16-bit microcontrollers; however, they typically execute more complex instructions (which generally means tasks take less time and energy to complete). This class of processor is emerging as the leading option for the IoT due to its ability to balance low power demands with the need for higher performance.

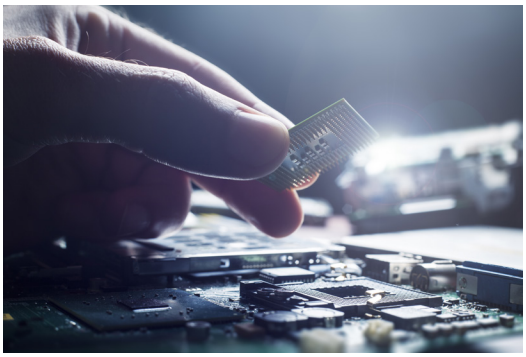

Figure 3. Choosing the right microprocessor for an IoT design can help you meet power, performance, and functionality requirements.

Within each of these classes of products are a wealth of options: different tradeoffs of power vs. performance, integration of peripherals including wireless radios, integration of security features, different memory sizes, different power architectures, and so much more.

Not All Microcontrollers Are Created Equal

Even within the same class of processor there are drastic differences in performance, power, and memory capabilities. In many cases, manufacturers make decisions to favor performance or power. In fact, wafer foundries offer different process variants that might help lower power consumption but, in turn, increase propagation delay, limiting the maximum speed of the processor. Some Cortex-M4F processors can run over 200MHz, while others cannot run faster than 100MHz, but they exhibit drastically different power performance. Some Cortex-M4F processors integrate 128KB or 256KB of flash memory, while others integrate 2MB or more of flash memory. In fact, integrated memories are one of the biggest cost drivers for microcontrollers, especially as the memories get larger.

Choosing an Intelligent Microcontroller

Searching for a microcontroller to add intelligence to an application is challenging with the myriad of products available. It is more challenging when you consider that connectivity means devices can be upgraded in the field; in other words, you can design IoT applications without really knowing what they will want to do in the future. While it isn't practical to consider every possible future upgrade, it may be prudent to make sure devices have some excess memory and processing capacity for potential upgrades.

For the universe of invisibly intelligent devices, 32-bit microcontrollers tend to make the most sense: they have a lot of horsepower while doing a good job of managing power budgets. The following are some key parameters that designers might consider when selecting the brain of their application:

- Horsepower: Is 48MHz enough? What about 96MHz or 150MHz? Remember that the faster processor might be manufactured in a process that is higher powered, so faster may not be better. On the other hand, if your application today is already consuming a fair amount of bandwidth in your processor, consider that you may need headroom for future application additions (what if you wanted to run a new algorithm in the future?) or adverse network conditions (what if the number of nodes increases so there is more traffic to process?).
- Signal processing and floating-point acceleration: Cores like the Cortex-M4 with FPU have acceleration for both digital signal processing (DSP) and floating-point operations—key for many algorithms that try to distill actionable data out of raw analog readings. Without these accelerators the microcontroller may need thousands more cycles to compute insights—while the accelerators consume more power when they are active than just the microcontroller core, they are generally only clocked when they are in use, and when they are in use they are saving significant time and power. On the other hand, these accelerators have high gate

Larger memories enable more intelligent things

counts and consume a fair amount of die size—but remember that modern microcontrollers continue to be built on more and more aggressive technology platforms, meaning that the cost adder of floating point and DSP support is less and less meaningful. Longer term trends in the IoT also support the need for signal-processing code to reside on IoT end nodes. The alternative is to send all raw sensor data to the cloud for computation, but this is a bandwidth- and power-intensive operation. If the devices can compute at least intermediate results (or even compute complete insights) the overall system power consumption can be reduced while also easing network congestion and reducing response time to data. Plus, remember our theme of "invisible intelligence," if you can't get the data and insights you rely on because of network delays, the intelligence is no longer invisible!

- Memory capacity: While embedded memory can be one of the more significant cost drivers for microcontrollers, the cost pales in comparison to the cost of replacing a device that doesn't have the memory size to handle future upgrades! Larger memories make it possible to build more intelligent devices: you can fit more algorithms and more decision-making in devices with larger memories. Having extra code and data space leftover give you room to add more features in the future.
- Memory expandability: When embedded programs get to 512KB or more, the number of microcontroller options with that much embedded memory dwindle. Even more so when the code starts to be measured in megabytes. But some embedded microcontrollers are capable of accessing external memories for both code and data storage directly from the core…that is, without moving blocks of memory from an external device like an SD card to the internal memory. Peripherals like an SPI Execute-In-Place (XIP) engine let cores run directly from external memory. While there can be a performance hit when running from external memory, these options at least provide a mechanism for applications to grow in complexity beyond the limitations of an embedded nonvolatile memory.
- Security capabilities: We'll talk more about security in another white paper in this series, but adding intelligence to devices usually means adding some special intellectual property or collecting special data. If there is value in the data, then there is likely value in the way you collect, process, or interpret that data. And if there is value, then there is the risk that someone can copy your device or use your data for their own purposes. Look for microcontrollers that can help protect your data and your intellectual property.

Bringing Intelligence to the IoT

Opportunities in the IoT are really a vast universe, and all devices won't require the same amount of intelligence. Adding intelligence to a toaster to get just the right level of crisp might not require a powerful 32-bit microcontroller because there might not be a lot more value to get from the smart toaster, but adding intelligence to your front door might prove to be extremely valuable since so many real-world interactions happen there: a 32-bit microcontroller surrounded by an array of sensors might help us monitor deliveries, communicate with visitors, and safeguard our homes.

There will be no one single solution for adding intelligence to the devices around us. But designers who create these intelligent devices should consider flexibility for the undreamt of uses of their device, and ensure that they design in enough intelligence for the future.

More Resources

: Also, read our related white papers

Powering the Next Generation of Smart Devices

Securing the Next Generation of Smart Devices

:For more information, visit

www.maximintegrated.com