Let's get to business here a little bit. Can you describe for us at a high level, what's the software that's needed in a sensor or a low-powered IoT device? What are different components? Just give us an overview of that.

If you look at the Micro controller space, we'll just bigger processors where we have four operating systems. But a Micro controller space, you have a few different levels of software that you can work with. You can do what we call as Bare Metal so if your drivers they might come with the device that you got from a silicon vendor or might be part of a project that you use so drivers are starting point. And then you may do your own software taken some communication standards that you need from various places immigrated and basically write your own main loop and truly just the basic level of engineering but a lot of really simple embedded products actually used to that. So Bare Metal driver based on embedded devices are out there.

Now that works for a small set of functionality you can only do so much with that and maintain it, you have also to maintain at some point. So the smaller Micro controller tends to use this. Another level for software for devices is what we call low-power operating systems. So where the goal is to get as much software functionality in the device as we can in a standard package reusable software but the goal of the operating system is not to do things like real-time scheduling and control but really just to optimize power how can we get this different function to working in a way that we're at the lowest possible power consumption we're using resources efficiently, we're not using more flash at RAM when we have to, you don't require as much clock speed...then that's another class of operating system.

So power is the fundamental driving factor in that case?

And functionality.

The power consumption.

Power in giving just the standards.

Balancing the two...okay.

Yeah, the different functions you need that you have reuse right that's the two major things. And then we have a third class of software for Micro controllers and that's what people know as a Real-time operating system. And real-time operating systems aren't really how we think about operating systems in the pc world for example and they're really just scheduler kernels, so it's just the scheduling part of an operating system. So if you look at Linux for example that would be the Linux scheduler that gives you your process. And those are really specialized in real-time control applications, safety applications and you do things like to scheduling where you can really make sure you got hard time limits on how long different pieces of software can work or how fast you have to meet a deadline for control. And those tend to be very specialized so you have different sets of software that come with for different
industries you might have specialized in automotive, some in more industrial applications. And so it's really fascinating class of software 'cause it really runs a lot of the hard industrial applications we have it runs you know, trains and it runs aircrafts, and a lot on exciting things that run on Real-time op.

**Planes, trains and automobiles.**

Absolutely, yes. It runs some of the main applications of real-time operating systems. And it's getting exciting in this space I think that's the cool thing about the embedded electronics and software space is that we really start to have more, more software available and I think it's only getting better. In the Real-time operating system space we already have a lot of really mature options you know, everything from open-source real-time software like to commercially supported source code from...for example Mitrium, or Express Logic great examples you know, you can get commercial software but you if get the source code and compile it you get support and you know, at ARM we work with all of this some partners you know, making sure that all the support's great drivers etc. as we do a very wide range of Micro controller technology.

**So what were the open source or the non-commercial drivers you mentioned?**

So, great example for example for open source is Free RTOS, so, Free RTOS is well better known Open source Real-time Operating System.

**Okay. And what about TinyOS is that fall in that classification as well or Contiki or...where do they fall into the spectrum?**

That's not the real-time operating system spectrum, that's not middle one this low-power operating system would function. So TinyOS and Contiki are a bit great examples of free community open source based projects with the roots back in academia. Now just recently from ARM we actually announced what we call mBed operating system, embedOS and that falls in that category as well so that's. In that, not real-time but the 25:00 really low-power devices were we focused on internet connectivity and security in particular.

**What are we calling these, low-power no, they are not OSes you said so what are we going to call this number two there?**

When things settle down and when you look back in time let's say you look back five years from now, we're going to call this things Operating system. It's simply as that right? It's just going to be an Operating system and we're not going to think deeper into it. It's going to be the whole set of software from drivers to scheduling, to security, to communications that you need to go start creating your application, right?

**Yeah. No, I'm talking about the metal...so, we got the bare metal you're writing your own drivers, your main control loop, okay, that's our...I'll call it the bare metal software. And the second one is for energy management...balancing energy management and functionality. And the third one is the RTOS and I'm sure you know, it's evolving from there, but what's that second one? What do we call those? I mean Contiki is one of those, and TinyOS is one of those...do they have like a generalized name I know they fall all under the OS umbrella but for that metal segment 'cause I think that's important?**

I think we'll really end up just calling them Operating systems. Simple as that Operating systems for embedded devices for IoT.
As supposed to the RTOS, is it contrasting to the RTOS which are more specialized in particular verticals and often and fortunately having you know, they carry around with them?

Well I think that would really be the difference right, you know, the operating system or you have all the software you need works for most things they're not connected devices in particular and then you have Real-time operating system which are exactly that right...it's a software stack specialized in real-time performance and you'll naturally have this application at a different verticals, like for, or safety applications. And so that is how people will talk about it in the future.

Now, so if we squint a little bit we can see with more with with everything else that the hardware is obviously getting more powerful, there's more memory available and it's becoming lower cost. So I think as we've seen in general computing we're seeing prices going down, we're seeing capabilities going up so I guess my question is...when does the general OS or will the general OS ever be able to consume or eat the RTOS? And you know my perspective and I've said it many times is all about standardization and I really believe one of the greatest frictions they are holding back IoT today is as lack of standardization at multiple levels specifically at the IP level but I see RTOS if we're talking about a certain IoT applications, it's really important that RT part of the RTOS is it going to be subsumed or consumed by the general OS or is there always going to be place in IoT for Real-time operating systems?

So I think there will be a place. I don't believe that the real-time scheduler operating systems will be consumed by a general operating system. I think what you'll find is that they'll start to work together at some point. Now we already start to see that like at ARM for example we collaborate with RTOS vendors and provide them with parts of the technology we have. We have something called the client which does all the upper layer standards for communications from secure http device management and we actually bundle that so that it can be used on RTOSes and on Linux for example operating system. So you can start to see this kind of combinations but we also see also the Linux space right in Linux we have basic Linux, we also have real-time variations of Linux.

Where you can add on a special scheduler to Linux is not used very widely but it is used in certain applications.

In IoT?

Not so much in IoT, but more on really hard core industrial full applications you'll see kind of industrial pcs running real-time Linux. And I think you'll see a similar trend happened in the Micro controller space where you have a general operating system like mBed OS and then the ability to plug in some real-time things when you need that.

Yeah. And I'm okay with that as long as 30:00 and I don't know if you were talking with the upper layers if you're talking about the communication stack but as long as there is you know, a common communication stack or I should say a communication stack that use a standards and so that we can then you know, start leveraging the economies and scale whether it's an RTOS based system, whether it's a general OS, whether...I guess not bare to the metal although maybe some of the drivers but is that what you were talking about, is that possible?

It is and that's we're going right now. I mean that's the big turn we're going to see in 2015. You know, last year was all about "Oh, we're going to get the standards, where are the standards on IoT?" hype was really on the opposite, right? And the answer I've had for a lot of people when I talked to them is that "We got the standards, right? We've
been working on this for ten years”. We've got low-power IPv6, we got really efficient binary web protocol, we've got security, we've got discovery, we now have object formats you know the so alliance recently published whole set of objects, semantic formats really nice and reusable. You know, we've got device management from Lightweight all open standards based on basic internet building blocks, we have them and use them. So I think last year, we really got that message through the people we seen alliances that come together that really putting those pieces together and OIC is an example of that kind of putting all building blocks together and say “hey here's different architectures were you can use internet standards” and I think it's a healthy thing to do. Now the challenge this year's how do we get those standards to people...now, traditionally embedded software hasn't work like that right? Embedded software's been more like “hey, here's a chip and here's some drivers”. You need to go and put together the pieces of your thing and we're assuming your thing's really special at so, the same package of building blocks won't work for everyone.

And so it's a been a very fragment that you have to go a piece together a security stack here, Zigbee's stack here and it's and hasn't really provided for a scale. So the big trend this year will be software ecosystem. So, all that software and all those standards implemented in a way that people can get hold of them...and that's kind of a problem we're trying to solve at ARM now focusing on. We got you know, we're shipping over 3 billion Cortex M that's our term for Micro controller chips with our partners, of course we don't make chips but all of our partners over 3 billion a year. How do we get all the standards for those chips, right? That's a huge number of devices out there we could just get the software technology that we want.

Yeah. I guess the software ecosystems are getting dependent on open standards obviously because you have to write something.

Yep. And you have to get services working with your devices and that's the other thing that will happen as we have the software ecosystems available...that standards are easily available to developers. You need that other part of the ecosystem which is the Cloud players—services that can host your data, operators that can go easily you know, build out on a network for your infrastructure, server platforms that can talk to your devices easily make a web page. And that's why actually just at the end of last year, we announced that we're expanding our partner program to cover the Cloud side of things as well. So we used the only partners with silicon providers you know, support their chips with the software make sure all the drivers are great but we've realized talking to people in internet wasn't enough, we've actually go work with all the Cloud players including players like salesforce.com we would normally think of right as embedded IoT but they are they are one of the largest business application platform.

Yeah. No, and I think it's an important point that you bring up is you know, and to certain extent what you're really talking about is bridging OT and IT and so, yeah we need the cloud to be using the same stacks, the same standards as the chips, as the software I should say as a software, as the protocols which we've talked about on the show before we're going to be speaking about more and as the OSes that we're talking about now and all the hosted by the chips because that's what the Internet of Things is it just goes beyond the operating technology of M to M or the specialized industries that we've talked about before it goes beyond the enterprise IT which gathered I guess more human based information or inputted in whatever format and that's really...I guess that's really important is that bridging between the OT and the 35:00 IT and I guess if you think about it, yeah, it makes logical sense that ARM and other players would start thinking
about the communication in more holistic manner and so that's what you're seeing this year I guess the broadening of division I guess a more practical use of the standards, bringing together the standards from these different alliances groups consortiums and then to enable a broader vision of what...I guess communication means for IoT. Is that a fair way to summarize it?

Yeah, in getting the software to people, right, so we'll be open sourcing huge range of communications standards this year. One of our main goal is actually getting a technology to developers for free and the whole sets of technology, not just the little pieces of it but things that can they really use on products.

Well. Sorry to talk about in particular now. Okay, so listeners to this show, we've got all types listeners to this show but although we're talking technology now because I really believe you have to understand how it works or you could put it to work really, you know the focus here is for business executives, leaders, managers that are thinking about implementing IoT within their business as a process or a product. So, okay, we're going to make our connected product. Now, we need to make a lot of different decisions I've spent again the first five episodes of this podcast on sensors in particular or the Micro level not at the embedded level we're going to do that as well. But what do they need to think about...when you're planning your IoT business, what do you need to think about from a software point of view, what are the important factors that you should be taking consideration when you're building both your business plan and your requirements doc? 'Cause I'm assuming that's where we let the serious techies, I guess like yourself, take it from there but from the business point of view what do we need to know so that we can talk intelligently, budget intelligently and make the right strategic decisions when it comes to the software on...yeah, I guess we've been talking holistically but in particular for this episode I want to talk about a software on that sensor...what are the considerations, what are factors here?

Well I think there's a couple things...first of all, you know, you can't separate the hardware from the software. You kind of have to think of them a little bit more holistically so first I think it's really important to choose platforms which have a really long life cycle because not only you're going to putting the device out on the field for a long time, you're also going to be evolving the product, you might open up new versions of the product, new variations as you succeed so it's really important to have scale around the hardware families you have.

But why do you need to separate hardware and software from the evolution that we've been talking about specifically if we're looking at standards and then this ecosystems theoretically they should run in whatever hardware that you choose or is that just not the case, just not that realistic to that.

For over the standards absolutely so if you're thinking about for example interoperability with another device even from another vendor could use...

I want...yeah, that's what I want different interoperability between different vendors but that does mean I have to pick a specific hardware or?

Not at all so that's definitely doable and that's something you keep in mind when you're choosing software is that you have this interoperability, you have the standards for that reason. However though when you're making the hardware decisions within your own organization for your own products.

The business decision in those case, just picking the right hardware that will last.
Scale, right? You want to reuse the same micro controllers, families across the same chips, you want to maximize your volume of hardware that you can actually purchase when you're making that across devices, you want to build or reuse radio interfaces, drivers. So that a lot you can do with a certifying modules for what you're doing and then reusing that across for products. So, I think, hardware in that sense is a business choice of how you create scale across your products, that's important to keep in mind. And where the software right is that well, when your choosing software ecosystem and your building at your software toolkit, you want to make sure that you can reuse the same software applications across those products, right? You might have three different variations of home automations sensor for example...you definitely don't want to have three completely different software toolkits, you want to have the same one and you may have three different variations of an application, right.

So is this big decision number 2: first one, picking the proper hardware, hardware platform. Second one, picking the proper software ecosystem?

Yeah and I think in particular making sure that you have a software ecosystem that can support the different hardware communication interfaces, applications you want to create. And as much as possible I think what you going to see happening is that a lot of budget used to go into kind of hand piecing together, lots of proprietor software. I think that's going to change. I think the budget is going to be more in the hardware, making sure you have the best possible hardware and design for your device, right, the importance of design is getting bigger and bigger. You know, embedded devices are becoming sexy they never used to be like that, right? Look at the thermostat.

It's pretty sexy.

It was never sexy, now they are, thanks to NEST. I think you're going to see a lot more of that, a lot of energy will go to design. And then the energy won't go into piecing together your crypto libraries for software that will just become part of the software ecosystem you chose.

Yeah, but from what time are we talking about Zach?

It's happening this year. It's going to be really exciting that's what gets me out of bed in the morning is that this years' going to be super exciting that we can actually get that kind of software to people so when they.

So software ecosystems, that's what I want to talk about. So software ecosystems, now, these are directly related choosing your software ecosystem, are these directly related to whether you're in the AllSeen camp, the Thread camp? The whatever camp or how they delineated and what are the choices they are available then?

No, that doesn't have to with standards luckily and that would get realty messy if it did. It's not about the standards...standards are the whole set of things that you need to have at a software, right? You need to have implementations of standards so whether they're you know, Thread, or WiFi, or energy, or in general all those need to be supported by a software, it's going to part of the software toolkit. Just like, those things are supported in Linux, right, out of the box you got Ethernet and WiFi support. TLS security for example...that's not something that can be an add-on right? It's going to be part of your operating system, part of your software...all of the standards we've talked about even part of the software that you use. So when it's about software ecosystem it's really about finding an easily available software where you have an ecosystem around it, you got enough people that you work with as a
value chain right...because now it's not just about you as a company you probably have to work with other suppliers, module makers, gateway manufacturers, gateway platforms...you need to work with service providers. You do work with maybe Cloud providers, right? Maybe you want to host your applications somewhere so you want to make sure the software ecosystems actually has those players involved so you can easily go work with them.

**So, software ecosystems are these now package as products are there more than one to choose from?**

*’cause I mean, I like the evolution as supposed to happened to think about it at the protocol level at an OS level, you know, different protocol usually after the gateway whatever...now your saying “no, you think about it as a software ecosystem”...great, that's make sense to me logical sense. Now again from a business perspective are there choices out there right now that we can actually make or what's the variety out there?*

So, we're just starting to see any evolution of this in embedded, right...we've had it in mobile right for a long time...great example you know, Android is a software ecosystem in a mobile absolutely, so is iO's actually in Cloud we have this for a long time different packets of technology that were created together software you can go get toolkits for free open-source. We're just starting to see it and one of the early ones on embedded was Arduino.

Arduino is a great example of a software go get the toolkit, go use it, you can prototype your stuff really quickly...I'm you know at ARM we're big fans of Arduino. That's one aim that maker kind of...prototyping face of evolitional product. Now what we're going to see this year's is the start of software ecosystems for actual products which used to be and that's really what we're focused on with the ARM mBed projects is actually doing that and that's what we're going to be bringing out this year is actual complete software ecosystem. So not only the software for free but also all the partners making modules, gateways, operators who actually host services for those (41m 54s) software providers that make sure things are compatible so that whole network of different companies.

**Now, I dig it and it makes a lot of sense but I want to have more than one choice. 45:00 Is it just that mBed is coming at first or there are others out there, what's the landscape look like?**

So right now, mBed is one of the bigger efforts going on we've had some software ecosystems from individual silicon vendors for quite a while, so different silicon vendors tend to kind of create their own software toolkits and that's been quite a while and you can do a lot with those. You will see things that are programming language specifics so you have things like JAVA for example...that's what Oracle does making sure that JAVA runs on devices.

**So it becomes an ecosystem?**

Because it runs on language, now, the lines are not so clear though right? So if you look at JAVA we're actually use mBed OS to run JAVA because you need an operating system under some JAVA interpretive language so that's actually working together. You may choose what language you write your programs in always be that it's one another and may be that one sits on top of another for example.

**So looking at it again from a business perspective, your recommendation is, okay, start looking more at Micro level, you look for if possible a software ecosystem that you can buy in to that's going to support the different use cases you have, the different hardware configurations that you have.**

**What other advice can you give? Again, looking at the software for the software decisions that have to be made...are there any other big ones?**
So I think another big thing is the changes that we don't have to invest as much, pieces as supporting software can see a lot more effort put into application. So the application that goes and runs on a device that's what our designers and the companies we work with, that's what they should be thinking from day one...how do we make the user experience great, right? How do we make sure that this device has the features that really know fundamentally changes the problem we're trying to solve if we're making a thermostat or a light bulb...what is it that we need to do to make that really great. And focus on that from day one rather than that building up all this nerdy technology as engineers we love to solve those problems but that's not the problem that the consumer has, right?

Okay, so in this case what you're talking about are the apps for the products and the products using sensors whether distributed or whether built...they are embedded within the products itself so you're not talking about writing apps on the sensors themselves or is this just a semantic kind of difference that we're talking about here?

So, yeah, you don't actually write sensors, apps on sensors per say, you write apps on the actual Micro controller but you have a lot of system on a chip technology being used already now where the line between this is getting very blurred, we have Cortex M cores built in to all types of sensors sometimes it's a single sensor, sometimes it's a whole range of sensors built into the chip itself and so it actually have processor plus sensors all in one package.

Okay, and so that will be enough to run your entire embedded product then is that what you're saying?

Absolutely, yup. Absolutely we're seeing devices easily with tens of kilobytes of RAM, hundreds of kilobytes of flash with very low costs being integrated with sensor socks. That's been happening already for couple years now as a trend of integration.

Okay. So for the sensors themselves, in the case measuring a process within in a retro fitting I'll say, existing process and from the software point of view again the recommendation is look at the software ecosystems and that pretty much covers at then, is that right? Because that's going to take care of the operating system whether it's real-time or not, that's going to take care of the communication stack.

Security, and you still have to do a lot of work, right? But then you'll be able to focus your work for example you may have a very specialized sensor that's the special thing you do, you might write driver for that sensor. And then you may contribute that driver back to the software ecosystem so that the ecosystem grows so a lot of the people involved do is their job might also be selling that sensor. And then you focus on the actual functionality of the device and you kind of shift you effort on that.

Now the other big trend I want to point is that, I love this quote, 50:00 is that for one of the analysts that we work with is that by 2018, over fifty percent of the applications in IoT will be from start-ups and not just any start-ups, start-ups that are only being founded now so they don't even exists as of last year. Half. That's how innovative IoT will be, right? That sounds a lot like the early web to me when web start taking off, huge number of start-ups totally innovating how we run businesses, how we run social networks, how we solved basic problems of government that didn't exists before they were founded to go solve those problems. We're going to see a different, a similar trend innovation in IoT and I think that's why it's really important when we build technologies, so from tech companies, we need to be building platforms for innovation not here's the vertical solution for do-me home automation and you must use it this way and that's it...more here's the toolkit of technologies we've figured out the basic stuff you need...
to go and communicate, do web communications etc. go build what you need to do and do it fast, we're not going to do anything that limits what you can create. And I think that's a paradigm is that we're seeing at the same time and that's what the driving software ecosystems is the need for people that just innovate. Bluetooth and Kickstarter right now you see tons of interesting projects we see them come across at ARM when they use one of the software they're going to tell us about like Oh, wow that's cool I didn't know you can do on just recently kind of little scale like a little flat board you put inside your fridge. And you put your different and things that you normally have in different places in your fridge and it tells you when you're out of something. You're running low in the milk department you usually have this much, time to go at the store like just a really simple I had this problem, I'm going to go solve it type of project.

I checked all at that because I would like just take but almost a year ago today when the fridge is among other consumer devices we actually set up as a button to send out spam but just check on that, it had to happen but I think your point is an important one from the business point of view, think about what you're deploying, generalize what you're deploying not to just solve the problem that to you are planning to solve because specifically, there may be other problems that your platforms can solve and if you're using something like an ecosystem that's hopefully using standardization which means that small companies and start-ups can actually innovate and that the fifty percent number that's assuming of course that there's friction that I've spoken about earlier is kind of eliminated and there are some standards cause you're not going to have start-ups taking bets on taking flyers but the business lesson here is when you're deploying, yeah, you may be doing a home automation and maybe a security but maybe think a little bit be a little bit more in the sense of the hardware that you choose but specifically the software you choose so that you can perhaps offer different products in the future that you didn't even think about or opening up your system for other people to develop on.

And now we have a whole ecosystem without silos. Other company making products, you can bundle theirs; you could sell your products into their offering and their regions. You can find new service providers, operators to go and sell your products in their markets all through business relationships rather than through all a lot of engineering and that's really exciting we're going to see that, t's starting a little bit now, we see some ecosystems of these different vendors working together I think in next few years you'll start to see more and more of that.

Halleluiah bother, I'm all over it, appreciate it. Tell our listeners where they can find about you, about your company and what are some date points that they can put in the show notes?

So I think the best place right now to look for information is on at mBed.org for what we're working on all the free software ecosystems I was talking about so mBed.org.