PROGRAMMING CONSTRAINED IOT DEVICES

PETER HODDIE VIDEO TRANSCRIPT

Tell me a little bit about yourself and your background in IoT.

I'm the Vice President of the Kinoma Software Team here at Marvell Semiconductor. I'm sort of a programmer by day and manager by night or vice versa. I try to balance those two so I still keep my hands in the code and know what's going on. I found my way into IoT through mobile. I was doing mobile app development and a lot of the same concerns of early mobile development in terms of constrained environment, not altogether reliable networks, and things like that were very similar problems to IoT. As we merged Kinoma from a standalone company into Marvell Semiconductor, IoT was sort of a natural transition for us.

Kinoma has a bit of a history. Obviously it wasn't started specifically for IoT but it sounds like there are similarities. Maybe you can give us a little bit of a background on where Kinoma has come from.

We started at the very tail of the Internet bubble working on Java software that's portable for embedded devices, which is kind of what IoT kind of become. We worked very closely with Sony for about 5 or 6 years and that really helped to set up the requirements for what we would need in a platform, what kind of technology. Some of the really core decisions we made, one of the things we learned from working with Sony is that there's always different divisions. The engineers from one division couldn't work with engineers or run projects from another division because the software stacks were different, the language was different, the tools were different. It was like all these independent companies but they knew they needed to work more and more together. There were these initiatives to get their products to talk. Part of our contribution to that was this idea of using JavaScript as the programming language everywhere. At the time, it was a little bit radical but it has certainly become almost accepted in the mainstream kind of idea today. That really worked. For example, at that time, engineers who had been working on portable media products – digital video players – and drop them into the camera division, they were very effective. We could take some of those same skills and apply it to the computer division of people doing the Vaio at the time. So this idea of being nimble with software by having some commonality in the software stacks is really a core part of what we've been doing at Kinoma ever since the very beginning.

Kinoma is an embedded chipset. Where do you see the business coming from, let's say in the next couple of years, for embedded chips just like yours?

Kinoma is really the software that runs on the embedded chips. Marvell, our parent company here, makes the chipsets. They make mobile chips, IoT chips, the chips that go in Google Chromecast. The thing that we bring is really unique that Sony started is our same application software runs across all of those. Once you've learned how to develop software using Kinoma, using our scripts, you are able to make something that works on a TV, or make something that works on mobile, make something

that works in a printer, make something that works in a light bulb. And that is really powerful. That's really important to our customers because companies are doing more and more diverse products and every time they start a new one, they don't want to have to learn something from scratch because the development cycle, which used to be 5 years for a product line and now it's down to like 6 months. They just don't have that time. So learning, being able to reuse the skills, in some cases being able to reuse the code is really significant benefit for people working in embedded.

I think also the development environment specifically wanting to have a familiar development environment and also something that my audience talks about a lot is support, the programming support. What support are we going to have? And therefore if you've got more of a broader platform then you can bring more support to bear.

Support's a big deal. One of the things that you've seen in the last few years in embedded really come in to play is open source. People are really demanding open source for any software that they're going to commit to because it's often not an issue of price but it's just kind of insurance. What's going to happen a few years from now? What if I need to make a change and you're busy? We see support in kind of three layers – we have open source where you can do it all yourself, you're good; we have online support forums where my team of engineers hang out and can answer questions to people directly; and for customers who are engaged in a commercial relationship with Marvell, we work very directly with them, 5:00 sometimes helping them to build out the architecture or optimize their code – things like that. Different levels but it depends on where our customers are at.

Independent of whatever your company work with, we're starting to see brick devices where there are companies that are no longer in existence, maybe they're not even not listening to you but they're not even there to listen to you anymore. That does actually bring up a point of focusing on software rather than hardware. Having said that, if you're using a particular development environment, I suppose there's not that much flexibility in moving it to a different hardware partner for example, is there?

That's part of where we're really trying to change. One of the reasons we chose Javascript as a language to base everything on is it was never tied to any particular operating system or any chipsets so it's intrinsically portable. By open sourcing our software, we already have it running on probably a dozen of different operating systems over the history of the company; people can take it to other silicon. Little by little, we're seeing that. People are taking their open source software and bringing it up in different places and I think over the next year, we'll start to see some significant deployments that aren't just on our silicon. It's some work to do that but it gives people a lot of flexibility to go where the cost structure is right, where it strategically make sense for them.

There's also risk management issue. Let's talk a little bit about developing for constrained devices. In particular, what I'd like to know is what are the biggest challenges, what are the biggest costs when you are developing for a constrained device?

The biggest cost tends to be the thing that you didn't plan for. People spend a lot of time trying to optimize the hardware cost. "We're going to save 50 cents on this chip; we're going to save a nickel on the sensor." When you get to a certain volume, that really matters but for most IoT devices, it's tens of thousands of devices and the place where all your money goes is your R&D cost, your software development cost. I always kind of advise people to do the hardest part first and for different companies, those are different things. Some people it's your product lives or dies based on its battery life. It's okay, don't make everything work, first just figure out if you can have this thing on and doing the very basics it needs to for however long that is. For some people, it's performance. If this thing doesn't respond in half a second, it's a fail. Can you do that? Because who cares about the rest. For some, it's memory. We have a million and one features and we're just not sure we can pack them all in. And so getting that sorted out really early will tend to minimize the R&D surprises down the road. It's never the same. For some people, it's networking challenges. You got to work in this very unreliable network environment. How do you make that work? Part of it is working with a platform where you can kind of see your way through to solutions to that; part of it is having tools. If memory is the issue, can you measure memory? Are there tools available that will show you your memory use clearly just so you can know when you're making progress because visibility is the first step to solving those problems?

That's a good point. Now switching, being very specific to IoT and obviously an important component of it is sensors. What I'm interested in is a big part of starting out a project, once you understand what data you need is then figuring how you're going to capture that data. Some of it is going to be on the Internet, through micro services but a lot of it is going to be through sensors. So when choosing sensors, will they work with any embedded system or vice versa? How does that work?

When you're looking at prototyping kits, there are two schools of thought. One is a very open school of thought; that one is things like Raspberry Pi is a good example, Kinoma Create and Kinoma Element are products that are both open in that respect so we kind of give you the pins and you can connect what you want. That requires a little bit more knowledge but it gets you access to almost everything. The flip sides are there are devices that have a specific set of sensors for them. The most fun example is littleBits because you can snap things together, literally a child can do it and they can't hurt themselves. It's great, the pieces are $20 or $30 a piece but if you're building one, it works well. It's a fun way to get started. There are things like Arduino that are kind of in between, which has some standard form factors like the shields that people typically use with Arduino. They work great and they are really easy to stack up but again, they sort of limit you in terms of form factor and what's available compared to more general, which Arduino can also support. My advice to people is if you're uncomfortable with hardware, something that just snaps together is a way to learn. "Hey, what are these things, what do they do, can I connect them?" 10:00 As you're getting into product development, you may try half dozen temperature sensors to get one that behaves the way you want because sensors are not as kind of automatically simple working pieces people imagine and so having a

platform that is open where you can just plug those things in is really critical to getting to the right selection of components for the final product.

I understand from the prototyping point of view, that makes sense. But at some point you're going from your prototype to moving to your minimally viable product, you need to be using something that's going to be commercially available that's going to be mass –produced. On those boards, will all boards and embedded chipsets be able to support pretty much anything or what are you looking for?

It's never that easy. I^2C, which a lot of the sensors are, is a good example of a standard that's not quite standard. In our development, we get as many I^2C devices as possible just to prove that they work because you do run into strange problems with some of them occasionally and a lot of devices tend to be a little bit more sensitive to voltage levels we've seen so that can be a challenge. Theory is these things should always work together. The reality is that they don't always. The Raspberry Pi as a good example doesn't really have built-in PWMs, which you need for a certain class of problems especially robotics. If you're going to do things that involve a lot of control, you would want to use that platform that has that, something like a BeagleBone for example is really good for that. We recently did some enhancements to our Kinoma Create to give much more control over PWMs. My real advice to people is not to just assume that data sheet is true but to really get in and try it because you're never going to know for certain until you actually put it together and put it through its paces and kind of the way you expect to use it.

So if there's a prototyping environment that uses the same hardware interfaces then that's probably a good place to start and if not, then after the prototyping stage, you're probably going to test it with a number of different embedded systems?

Yeah.

Let's talk about cost now. Put things in perspective for my audience in terms of what are we looking at in terms of cost? Not the sensors obviously but more the brains that are on the things themselves.

There's a real range and it depends on what you're looking for in terms of performance. Adding an Arduino to a product is a $1 or $2, it's great. As you get in to more and more performance, you can be up into the $10-$20 range. Google Chromecast is a good example; it's a $35 product. That's not a high margin business for Google; they're doing that to seed the market. You've got to figure out what's important. There's starting to be some all-in-one chips, which save some money but also just save board space in engineering. The Marvell part that we've been working a lot with recently is the IoT product from Marvell called the MW302. It's one chip the size of your fingernail that has 200 megahertz ARM processor in it, Wi-Fi built-in, and half a megabyte of RAM, which turns out to be enough actually for a huge fraction of products out there. So you basically bring your power supply, a crystal, and flash and you've got the ability to Wi-Fi enable any product. There are other people who do similar things, for example with BLE. And I think these all-in-one things are great because it means that the chip manufacturer has done the pre-integration for you. It also is, in general, going to

consume less power, it's going to consume less board space, it's going to save the engineering effort of wiring everything together. I think you're going to see more and more of that. It's kind of the SoC for mobile that had the GPU built-in but the networking is really a core part of it, the memory is there. That's kind of a traditional part of embedded as well. Those things are, by the time you get a board together and everything tested and whatever, still single digit dollars, say less than $5, maybe substantially depending on exactly how you do it to Wi-Fi enable a product. And that's only going down. I think it's very cool and it's why IoT ends up being inevitable. If you look a couple of years out to a three years out and it costs $2 to add Wi-Fi to your product and that means it's remote control, reconfigurable, and can connect all these things, you're making a product that's a $100 or $200, it's going to start to become just something you're always going to have. And by the way, that can be the microcontroller for everything else you do 15:00 on your product and you probably need one anyway. And so IoT really just becomes the part of the woodwork. How we use it, what the scenarios are, how we enable an Industry, how we enable cities, how we enable consumers, it's still an open question and we're really all actively exploring that but the price point and the capabilities of that price point are really phenomenal as you just look a little ways out.

Where can my viewers find out more about you and Marvell?

The Kinoma website is a good place to start, www.kinoma.com. We really are big believers in open source, open cloud, open sensors, so you can see all of that there. There are tons of sample codes, tons of documentation and of course it's open source, so almost all source code is there as well so people go and check it out. To get in touch with me, Twitter is a great way to do that. It's a quirky Twitter feed but aren't they all? It's @phooddie on Twitter or the Kinoma Twitter feed, @Kinoma.