EPISODE 681

[INTRODUCTION]

[0:00:00.3] JM: Internet of Things is a concept that describes lots of devices that you interact with regularly, being connected to the internet and networked together. Technologists have been dreaming of this world of IoT for many years, where our connected refrigerator can detect that we're out of food and automatically order food, or our connected bathroom can scan us for diseases and recommend treatments to us.

The bright future of IoT is slowly coming together. Hardware prototyping is getting cheaper. Voice interfaces and machine learning are creating new mediums for communicating with devices. Platforms like Kickstarter are helping developers to validate the market for their products and raise the necessary capital to build their product.

Android Things is a newer developer platform for IoT applications based on the Android operating system. Android Things consists of hardware devices and software tools that reduce common IoT problems, such as software updates and security patches. Wayne Piekarski is a staff developer advocate at Google and he joins the show to talk about the state of IoT and why Google built Android Things.

[SPONSOR MESSAGE]

[0:01:29.9] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more, people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

© 2018 Software Engineering Daily

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily.

Thanks to DigitalOcean for being a sponsor. The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW]

[0:03:36.9] JM: Wayne Piekarski, you are a staff developer advocate at Google. Welcome to Software Engineering Daily.

[0:03:41.3] WP: Hi, thanks very much.

[0:03:42.4] JM: I want to talk to you today about Android Things, which is the things refers to the Internet of Things. Internet of Things is something that people have been talking about for a long time. Some people will say it's been five years away for 50 years, or it's been one year away for 10 years. In any case, many people think that we should have more things that are internet connected today. We do have a good number of things that are internet connected, even just our smartphone, the device we use most in the real world is arguably a thing that is connected to the internet. Where do you see the current state of affairs and where are we going in terms of the Internet of Things?

[0:04:25.1] WP: It's interesting you brought up smartphones, because computing is appeared and people don't even realize that it's become part of their life and they're using the internet and

stuff like that every single day of their life. It's just happened and no one really noticed. There's actually a lot – I mean, my thermostat has internet in it, lots of different devices in houses now have that stuff. We have Google assistants.

It's everywhere, but at the same time there's still a lot more that it could be doing. We're not quite living the sci-fi movie dream yet, but I think that's why we're talking to developers about Android Things is we're trying to show our platform to make it easier for developers to build IoT devices. Because in the past, it's been challenging for developers to get this right, to get security right, and we want to make that as easy as possible, which is going to enable smaller companies to make devices.

[0:05:12.9] JM: One thing I think about is I think a lot of it has to do with you'd like certain devices around your home to be internet connected, but the internet connectivity to them is not so powerful, at least in an obvious way, that it makes the device so much better that you want to go out and buy a new refrigerator immediately. It takes a really long time for a refrigerator to depreciate as an asset compared to a smartphone, for example. Relative to the best smartphone you could have, your smartphone a year ago is not that great, so it depreciates quickly. You buy a new one and all of a sudden it's much better.

Whereas, okay I can go out and buy a smart refrigerator tomorrow and I could probably get it into my apartment if I could somehow convince the management to let me install that refrigerator, but that to me and you could you could say the same thing about my TV, or my mirror. Seems like the value ad for connectivity is not quite there yet and the depreciation of those things that we want to be Internet of Things devices is slow. Would you agree with that thesis?

[0:06:17.6] WP: I think we're in a learning phase. We're still trying to work out what we want to do with these devices. Not every device should be internet connected, but you look now people are carrying wearables around. Now these wearables have heart rate monitors and fitness things. People are using these things, because it solves a problem that they couldn't do before. Now you have the ability to control your lights. You can turn them on and off from home, you can tweak your air conditioner. There's lots of different places where it could be and I think we're still in an exploration phase of where we would put it.

Now up until now, IoT has been quite expensive to develop for. You had to be a company who made fridges, or cars, or something like that to be able to afford the money to build some expensive IoT device. What I'm really interested in is what could a developer at home build, or what could a Kickstarter project build, or something like that, but not have to own a factory that can make millions of devices?

For example, when I start working on IoT at Google actually, I always have a project that I work on in my own time to learn about something. I was obsessed with making this internet-powered dog feeder, so that if I was late coming home from work, I could press a button and it would feed my dogs and they could eat something. Now so when I was working on Android Things, I would use that as my case study of how to build something like this.

As a single person at home, I was able to make this device and I bought a commercial pet feeder. I pulled the electronics out, put my own electronics into it, but at the end of it I built this really nice device that's like, wow, using something like Android Things, I could build this device myself and sell it in small quantities and be able to afford to sell it and maybe start a little business around it.

I think that's what's going to get interesting, because these mega corporations don't necessarily know what cool devices people want, but the people listening to this podcast right now they might be like, “You know what? I really wish I could build this device, because it would solve my own personal problem, but it's too expensive to get in a business. If only the cost could be lower, then I could build something.”

In the case of cellphones for example, for many years the phone companies controlled them and they didn't allow developers to make apps for them, but once the app door got opened up, everyone was making apps. Now every app that could ever exist has been created. Imagine if it was like that for the IoT space where anyone could build any crazy device that they've been dreaming of their whole life, and I think that's for us is getting very exciting is that the cost of development are going down. With Android Things, we're building these hardware modules called SoMs that you can embed into your devices where we've taken all the hard parts of the electronics and we've put them in a single little board that you can buy in small quantities at

good prices. That's where I think it's going to get interesting. I think it's these small developers, small companies are going to be able to make things that previously weren't possible.

[0:08:57.0] JM: You talk about the lower barrier to entry for prototyping hardware. As I understand, that is due to a number of different trends. One trend is that as smartphone production scaled up, there was just more and more – a more efficient supply chain and a more distributed supply chain of different manufacturers competing against one another and they're making these modules at scale, things like GPS devices that were now much cheaper to buy. If you want to have an IoT device with a GPS device, it's all of a sudden cheap to prototype that.

Are you seeing an increase in the number of dorm room hackers and startups that are just prototyping things and they're able to do it on a really low budget? Are you seeing a significant increase in these kinds of people who are doing "IoT?"

[0:09:49.5] WP: Yeah. If you look around on websites, like HackStore for example, we have a community on there where we encourage people to publish their projects. You see, all these people are building these little home gadgets. They're based on Arduinos, or Raspberry Pi's or BeagleBones, or whatever. There's a lot of these boards and developer kits floating around that allow people to make interesting little prototypes at home.

The problem is that they don't scale into production in that once you've built something around it, there's no real path where you can mass-produce that device and get the cost down to sell a real product. One of the other things, you mentioned modules in that you can buy GPSs and different things. One thing that's always been hard up until now is buying the CPUs and the GPUs that go with them.

For example, we have partnerships now with MediaTek and XP and Qualcomm. What that means is that we buy the chips from them. We've negotiated with them a design for Android things that we've built this module around. We've mass-produced these modules so that a developer can buy them. Now previously in the past, if you wanted to buy a CPU for a MediaTek, or Qualcomm, you'd have to buy a million of them, or a 100,000 chips before they were even able to talk to you.

Now with Android Things, you can buy one module, prototype it, and then when you're ready, you can then buy more of them in whatever quantities you need. What we've done is we've negotiated a design with these CPU designers, so that you as an individual developer don't need to do that. That was one of the big roadblocks for making IoT devices based on ARM processors before is you needed someone else to do this negotiation and write the kernel modules and all this other stuff. That was where things got really complicated.

That's a big change for the Android Things, which I think is going to make it a lot easier. You can buy one of these modules, drop it into your design, build a prototype around it with a developer kit. Then when you're ready, you basically take the developer board away and replace it with your own custom printed circuit board. One of the other breakthroughs that's happened recently is there's a lot of these printing PCBs on-demand now services, where you can send them a PCB design and then they have the ability to print you a small number of boards, ship them to you, pre-soldered and then you can drop your SoM in and test it.

It's very interesting in that you don't need to make your own PCBs anymore and there are these services that can make them for you, where you can make a very nice design. It's almost commercial quality that you could sell.

[0:12:13.7] JM: I've been inside a factory of one of those PCB places and I did an interview with somebody who runs with those companies called tempo automation, which people can listen back to if they're interested in that process of scaling up the circuit board production. One thing I want to ask you about is another overloaded term, which is edge computing. With IoT, when somebody talks to me about IoT, the conversation we end up having 80% of the time is of this connected home, or perhaps my wearable computing, but another 20% of the time they're talking about perhaps sensors at an oil rig, or distributed through an agriculture large-scale agricultural farm. Maybe they're doing soil monitoring.

Although more and more these days I'm hearing the term edge computing associated with those more industrial applications, things like agriculture tech and oil rig sensor manufacturing and production and software engineering. Do you think of the development paradigm for these two classes of engineering as the same thing, or are we talking about different types of applications here?

[0:13:27.0] WP: Well I think in the past, a lot of people have thought about IoT is light bulbs and thermostats in your house. Those are very simple devices. These devices have been around for a long time. You've been able to buy them for maybe five or some years. These devices previously were very limited. They had the ability to turn on a light, or detect a button press and that was it. There's not much computing happening there, apart from sending some network packets over Wi-Fi.

Where we're moving now is that with things like Android Things and other platforms is you have very powerful CPUs that are relatively cheap that you can embed into a device. You could have a smart camera that's running machine learning like Tensorflow locally on the device. When it sees something of interest, it can go, "Ah, there's a person in your house. Maybe they're not supposed to be there," it can then contact the cloud.

Rather than uploading all the video to the cloud and processing it in a server somewhere, you're doing the detection locally on the device and then only talking to the cloud when something interesting is happening. That's edge computing where you're doing more of the processing on the device itself.

What we're seeing now is that these new devices are very powerful. You can do this processing locally and then only talk to the cloud when needed. That's going to open up a bunch of interesting applications that people haven't thought of before, because as I said before, has been thinking of light bulbs and thermostats, but now there's all kinds of other possibilities. Imagine a audio processing system that listens to noises and detects when your dogs are barking at home and then flags that or something, or something that listens to the sounds of noises outside, or airplanes flying over your house or whatever.

There's all kinds of interesting video and audio processing applications machine learning that you could do locally. I think we haven't really scratched the surface of what's possible in that area. I think when developers start playing with things, like Tensorflow and whatever, they're going to be like, "Oh, yeah. I could build this really interesting idea that I couldn't build before."

As part of what I do, our job is to provide the tools to developers so that they can then take it and build their amazing idea. Yeah, that's where I think things are going with edge computing; lots of processing on the device and uploading to the cloud when needed.

[0:15:37.0] JM: You've talked about Android Things a little bit. Android Things 1.0 was released around Google I/O 2018, so very recently. Explain what Android Things is.

[0:15:49.6] WP: Yes. Android Things is our platform that's based around a concept of a SoM, or a System on Module. A System on Module is a prefabbed module that contains the CPU, memory, Wi-Fi, Bluetooth, the GPU. Everything that's necessary to run Android Things, we build into a single little module and then it has a breakout connector on the bottom that connects it up to a breakout board that exposes USB, SPI, I2C, GPIO pins, different things like that. We have a number of different developer boards that you can plug into to test out things, but then when you're ready to build a production device, you take the breakout board away and you build your own custom PCB and this allows you to embed the SoM into your device.

Now on the software side, rather than developers having to take their own Android AOSP and compile the code themselves and then put it on the device and bring up the platform, we at Google are building the Android Things software to run on these devices. We've worked with Qualcomm, MediaTek and then XP, and we have built software that runs on these devices out of the box.

As a developer, what you do is you use Android Studio, you build an APK, which is exactly the same development model that you'd use for a phone, you build an IoT application in Android Studio, you upload it to our developer console, it then builds you a firmware image that you can install on to the SoM. These devices are now your devices out in the field running whatever software you want on them, doing video, audio processing, machine learning, whatever you want. It's a great way for developers to build their own embedded IoT applications.

It doesn't even need to be for IoT. You could build your own IoT cash register, kiosk displays for an airport. There's lots of different things it can do. We've seen a lot of developers in the past using Android, where they've taken the Android AOSP and they've customized it for their own needs. That then requires them to maintain this customized version of Android. Every time a

new version comes out, or a new security update comes out they needed to take AOSP and update their own tree with all the latest patches and keep an eye on any security updates and take care of that.

With Android Things, our team takes care of maintaining those patches and making the firmware builds for you, so you as a developer all you need to focus on is building a single app that runs on your device, and this app runs within the confines of Android and you can use all the standard Android development tools, the UI, the user interface toolkits that we provide, things like that, all of that works.

An Android Things device is basically a phone without a screen on it, except you could also have the screen if you wanted. It's like Android for phones, but you don't have to buy a commercial phone. You could build your own custom hardware that you can embed into whatever it is that you're trying to build. That's what Android Things is about.

Then finally, one of the big things that's important about it is security. A lot of developers when they build their own custom IoT solution, they'll take Linux, they'll put a Python script on there that runs as route that's writing and reading from all kinds of places in the file system. If anything goes wrong, or if anything's exploited, someone now has control of the entire system and they can do whatever they want. Whereas with Android Things, every single app runs within its own little app partition – sorry, app container.

What that means is that even if someone exploits the APK that's running on the device that you put on there, because it's running as a separate user account, there's also SELinux rules enabled that stop you doing things you’re not allowed to, the system is very siloed so it's very hard to break out of the security system that we've built.

We take care of the security for that, we also provide monthly security updates to your device via our console. Literally once a month, these security updates come out and you can check a box in a developer console that says every month automatically update my devices. You could be a small startup, like a company that's making internet dog feeders or whatever and you have the ability to push updates to your device without having to maintain your own huge security team, or encryption team or whatever.

We take care of all these things for you, so we've got secure boot and verifying that the firmware downloads are up to date and not tampered with. We also have our own over-the-air update mechanism, so that you can push updates out to all of your devices and we provide it for you. It's actually the same update mechanism that Chrome OS laptops update with a Google.

[SPONSOR MESSAGE]

[0:20:19.5] JM: Logi Analytics is an embedded business intelligence tool. It allows you to make dashboards and reports embedded in your application. Create, deploy and constantly improve your analytic applications that engage users and drive revenue.

You focus on building at the best applications for your users, while Logi gets you there faster and keeps you competitive. Logi Analytics is used by over 1,800 teams, including Verizon, Cisco, GoDaddy and JPMorgan Chase.

Check it out by going to logianalytics.com/datascience. That's logianalytics.com/datascience. Logi can be used to maintain your brand while keeping a consistent, familiar and branded user interface, so that your users don't feel they're out of place. It's an embedded analytics tool. You can extend your application with advanced API, so you can create custom experiences for all your users and you can deliver a platform that's tailored to meet specific customer needs. You could do all that with Logi Analytics. Logianalytics.com/datascience to find out more.

Thank you to Logi Analytics.

[INTERVIEW CONTINUED]

[0:21:46.7] JM: I'd like to walk through the development workflow once again, because my sense is that Google's been developing hardware devices for a while now. They've also interfaced with a giant developer community, the Android developer community and this workflow of deploying software to a dedicated hardware device feels that is somewhere in between the two.

You're giving this this world to developers that is going to seem probably a little bit foreign to some of them, because it's not exactly like smartphone development. It's very similar, but I just like to walk through an example. Maybe you have an example from somebody in the community, or perhaps yourself with your internet-connected dog feeder. I know you already gave the intend example, but through the lens of a specific application what you did and just what the development process is like.

[0:22:45.7] WP: Okay. I guess, just so – let me talk about what Android's development process is like for a phone quickly. When an app developer wants to make an app for Android and iPhone is very similar as well, but on Android you run Android Studio, you write Kotlin, or Java, or C++ code within Android Studio to build an app. There's a user interface toolkit that you can build user interfaces very easily. Then you basically hit the compile button and it generates what's called an APK, or an Android Package File.

Then that APK file you upload to the Google Play Store and then people can then download your app from the Google Play Store. That's how basically phones – the phone ecosystem operates. Now on the side, the company who makes your phone also has the ability to push firmware updates to your phone to patch security issues that pop up. That's the phone ecosystem.

Now if you wanted to build the IoT dog feeder example, firstly you need to build all the mechanics of the dog food – of the dog feeder. You would buy either the shell of a dog feeder, typically it has like a motor inside of it with a little paddle that dispenses the food. You basically need to apply five volts to a motor to make the wheel turn and spit some food out. You would buy one of our SoMs and you would build a PCB that you would design yourself, using EAGLE or KiCad. You could use a prototyping board initially to make it easy. You would then hook your motor up to the board, and then in software you would use Android Studio to write a small Android Things app that processes this.

The way it would work is that you would listen for commands via the network. Whenever one of those commands arrives that says feed the dog, you would then write some code that says when this command comes in, raise the GPIO line for one second. Then the paddles will turn

and the dog feeder will dispense some food. Then at the end, you could send a message back to the network saying the dog was successfully fed.

You write this code in Java, or Kotlin, or whatever you want and then you would then build an APK in Android studio and then you would install it to your device locally and test it and see how well it runs, make sure it's all good. Now that you're done, what you would do is you go to the Android Things console and you would say, “I want to create a new device type. It's called Wayne's Dog Feeder.” You would upload the APK for it, you would pick the device type that you're using. I may have picked the SoM from NXP Semiconductors, so you would check that.

Then it'll build you a firmware image. That firmware image contains your app, plus the system image itself. Then you would take that firmware image and you would install it to your device, but then you would also – when you're ready to go into full production, you would send that firmware image to the factory where these things are being made and they would flash this firmware image onto every device that's being shipped out, so now that it's your device.

Now these devices would be sold to consumers and they would use them. Now if you want to do a fix to make a change to your software, or maybe add some new features, so you might want to make it so that whenever the pet feeding cycle finishes it could play a small sound, you would add that change to the software, you would make a new APK and then you would upload the APK of the developer console. Then when you press the deploy button, it will then roll that out to every device that you've ever sold, including new devices that are being turned on they would automatically get that update immediately on boot.

Now you've updated all your devices in the field and this all happens very easily and seamlessly. You didn't have to build your own OTA mechanism, you didn't have to learn anything new either. Any Android developer who's ever written an app for an Android phone could basically get started on Android Things pretty much immediately, because everything's exactly the same. The only thing that's different are the APIs to do the GPIO calls to manipulate the outside signaling lines, but the APIs are very similar to standard Android.

© 2018 Software Engineering Daily

There's a lot of companies out there who are using the Android right now, so it would be very easy for people to get started. That's the workflow for it. It's very easy. All the hard things are taken care of and you can focus on making what it is that your device is about.

[0:26:45.2] JM: Updates are something that's important to focus on here, because as you know, there's all these examples of security problems across Internet of Things device. There's of course the Mirai botnet, which was this situation where you had a component of a camera that was used in lots of different devices. It was a camera component from a bunch of different manufacturers, and there was a Linux – it's the Linux little board I guess on this camera, and the Linux had the same default username and password so that it was easy to just scan the internet and find a bunch of examples of these Linux operating systems that have the same login credentials, and there was a botnet that was essentially made out of these security cameras, out of these cameras. That's one example.

There's also the examples of the internet-connected teddy bear that has some security vulnerability. It's just very clean. Then you hear people like Bruce Schneier saying – and Bruce Schneier is a fairly libertarian guy, but you hear him saying something like, this is something that the government even needs to get involved with potentially, because we really need to regulate Internet of Things. We need to get some sense of standardization around that.

We don't need to go into that, but needless to say having some standard system of security updates for the Internet of Things devices that if I'm a random hacker but I want to build a business out of Internet of Things, I want to be able to handle updates, I want to be able to do security really easily and I want some part of that taken care of for me, even that would be great. Just I can trust that my Ubuntu image is going to be periodically updated, as long as I have it configured to the default settings basically.

Talk a little bit more about that. Why are updates so important and why hadn't this problem really been solved before? Because this seems like one of these key components in what Android Things is doing that’s novel, it's here's an Internet of Things platform where you can get reliable updates. Is that a completely new thing?

[0:28:58.6] WP: Well, in the phone space we've been dealing with this for a while. Phones have been doing updates constantly to fix up security exploits as they happen. Security is one of those things that I think a lot of people are now finally starting to wrap their head around. In the past people are like, “Well, no one's going to attack my device. It's a small device. Not in use that much.” Or they're like, “No one's ever going to guess the hard-coded password that they've put in somewhere.” There's been a lot of simple mistakes that developers have been making, where they're in a hurry, they take some random Linux distribution, they hack in a bunch of things to prototype with and then they forget to take them out when they're ready to make their production device.

They're in a rush, and then once they ship the device they forget about it, or the company disappears. There's all these kinds of – there's just a lot of bad practices that have gone into making these devices in the past, that people are gradually starting to wake up to. I think also there was no real commercial incentive for security either, because most people didn't think about security, so therefore consumers aren't asking about it. I think now and the media people are starting to see the impact of these security problems and they're like, “Well, I want a secure device. I don't want people logging into my camera in the middle of the night and seeing what's going on.”

People are starting to think about that and ask about it, I want security. Then device makers are like, well, you mentioned a few examples of some botnets and things. They're like, “Oh, we don't want to be part of that.” People are now waking up to it. Security comes in many parts. Firstly, it comes in, well don't do silly things like baking in default passwords and things like that. The other thing is a lot of these devices were leaving piles of ports open by default. I mean, they're leaving telnet and remote debugging ports just open, and running the Nmap port scanning tool, you could see these ports open.

I remember years ago having a router where you can Nmap and you saw the telnet port open and you're like, “What are you doing with that?” It's like, “Why would you even leave that open for?” I think security by default is very important. Don't have any ports open, unless you absolutely need them. Don't have any ability to use a default password, because you don't want to have to need that.

The next thing is that when you do have a security problem that you need to fix, you need an ability to actually do the update. Now a lot of devices actually have no mechanism, whatsoever to do an OTA update, over-the-air. What they rely on is the consumer putting a firmware image on a USB Drive, plugging it in, rebooting it, doing some magic steps. No one knows how to do that. There's a few technical people, but your average consumer doesn't think about it, doesn't want to do that, makes it too hard. A flash drive-based update mechanism isn't going to cut it either, but a lot of companies are doing that still.

Then there's been cases where some devices, it was physically impossible for them to be updated. Once they were flashed at the factory, that was it and they could never be touched ever again. That's just a terrible design choice that you can never fix that device ever. With Android Things, we've provided OTA updates by default. They're easy to do, so you can just make a new APK.

Also by default, Android devices actually do this in general where in order to get your certification for any Android device, whether it's Android Things, or an Android phone or whatever, the device by default shouldn't have any ports open with random login capabilities, or anything like that. That stuff is just not allowed by default.

Then next thing is that Android itself has a very siloed security model. On a regular Linux homemade IoT device, you typically run everything as root and any app has the ability to do whatever it wants, including rewriting the file system, or changing hardware things or whatever. With Android Things, what happens is your app that runs on the device if it wants to manipulate a GPIO line, it has to make an API request that goes via our binder IPC call to another process, and that process is responsible for changing the state of the GPIO pin.

That separate process has the necessary privileges to do it. The system is set up so that your app can't do bad things, because it's been locked down by the permissions that's been set up for it. These are the same permissions that are used on Android phones as well. It's designed to make sure every app can't interfere with any other app, and also interfere with the rest of the system. Security is not just one thing, it's built from the ground up. You've got to have siloed processes, you have to have an OTA mechanism to push updates, the system itself should have

security in place with things like SELinux and permissions to make sure everything is operating as expected and that apps aren't doing things they're not allowed to.

In Android, if you want to make an app that does audio for example, you need to request an audio permission and the system then checks that that permission has been granted. If you build an IoT dog feeder and you didn't declare the audio permissions, it's impossible to open the microphone to record audio, if you didn't request the permission for it. It's all about security in-depth, not making trivial mistakes.

The Android security team, I mean, they've been building Android devices for 10 years and doing updates to phone makers. Phones are constantly under attack by hackers everywhere. The Android team over the years has built up a lot of best practices and a lot of expertise in making secure devices. Now we're trying to leverage that to make IoT devices with Android Things.

It's all about many different things. It's not just one feature that protects you, but there's many features that make it secure. Actually, one other thing is that the actual update system itself when the updates come in, they're cryptographically signed to make sure they haven't been tampered with along the way. They're also checked every time the device boots up, the bootloader checks that it hasn't been tampered with as well.

There's a lot of different features that are in there to make sure that when you sell a device to someone, when that device phones home to your cloud, you can know that that device is a device that you've manufactured, it's authentic, it hasn't been modified behind your back and it's designed to protect against the common mistakes that developers are making. Because I think now, we've moved into the point now, like you mentioned, there are people calling for in, “Oh, we need rules about how these devices are manufactured and we need people to think about development properly.” I think it is very important that developers do think about security.

In the past they've been like, “You know what? I sold my device. I'm done. I don't need to worry about this device anymore.” I think people need to realize that when you buy an IoT device, you need updates for that device. You can't just build and forget about it. You need to maintain it. If you're a company making devices, you might charge a subscription fee to provide updates, or to

make sure the device is maintained. I mean, I've seen devices that have subscription models, where it's a pool monitor and then they sell you the supplies to treat your pool water with or whatever, based on an IoT device.

It's not just about buying an appliance then forgetting about it, but it's these devices need maintenance. It's like when you drive a car, I mean, you can't drive the car for 30 years without doing a service every so often and checking its breaks. You need those things for safety. With IoT devices, you need to think about maintaining it and keeping it running and what's your long-term plan to support your customers.

[0:35:56.0] JM: On the software side, Android Things is using Android. What are the differences between Android Things, that distribution of Android and Android itself? What's the model for example, getting changes, that there's some change to Android core, how does that get pulled into Android Things? Give me an understanding of the differences in code base there.

[0:36:21.2] WP: Yeah. Android was originally developed as a phone operating system. Then in the honeycomb era many years ago, they came out with tablets. Tablets was another platform. Then a couple years ago, they launched Android Wear and Android TV, which were variants of Android designed for televisions and watches and then we had Android Auto and now we have Android Things.

There's this core Android platform. Then over time it's – there's been many different variants of it for different platforms, depending on what you need. The basics of them are all the same though. They still have the same underlying libraries and UI frameworks and video and audio libraries and so forth. As a developer the APIs are identical. There's one Android API surface for all devices.

The difference is that on a Android Things device, you can't make phone calls and you're not going to browse the web, because it's not a device for consumers. A phone browsers the web, a phone makes phone calls, but an Android TV device doesn't make phone calls. On Android TV, they pull the phone functionality out. The same for Android Things.

Each version of Android is a customized variant, where some parts have been removed. In the case of Android Things, some parts have been added, like the ability to talk to extra peripherals via GPIO, or I2C, or whatever. There's this common Android codebase and the Android security team maintains that common Android codebase, so they're constantly fixing up bugs and patching security vulnerabilities. When one of those gets fixed in Android the main tree, all of the different Android products get those fixes applied to their distributions as well.

Every year, typically Google releases a new version of Android. Just recently, we released Android Pi. Once that is released for phones, Android PI then works its way to all these different products like TV, auto wear and Android Things. The code basically flows onto each of these sub projects afterwards.

As I said, it's basically the same Android on all of them. It's exactly the same development experience. Now in the case of Android Things, we customized Android to make it more suitable to running on our SoM modules. Android has many features in it that you don't need in an IoT device. We've removed notifications, we removed phone calls, you don't have a web browser, or the ability to download apps from the Google Play Store. These are not things that you want to do on an IoT device.

We took away things like that that aren't necessary and it also helps to reduce its memory footprint, reduce the amount of disk space that's needed, and we then build a distribution for that device. We also have things like Google Play Services on Android Things. A lot of developers actually want Google Play Services to do things like improve location tracking, or the ability to talk to Firebase and other different APIs like that. Android Things includes Google Play Services, so that you can do things like that.

Android Things devices actually meet all the Android CTS requirements, so that it can have Google Play Services on it. Where in the past, you couldn't do that because IoT devices didn't have a screen and the CTS tests required a screen. Now we've added all that to the certification process, so that these devices when you buy one of our SoMs, they're ready to run all of our software on them and it makes it really easy to make that happen.

It's a customized variant of Android that's designed just for IoT devices, but we've removed the parts that were from phone-based parts of Android that aren't necessary. The cool thing and a lot of – one of the big reasons that people use Android, rather than using their own homemade Linux distribution is a lot of them use Android because they want the user interface framework, or they want the video playback, or the audio framework or whatever.

Android Things provides all of the same tools for things like that. You can go to Android Studio, there's a drag-and-drop editor where you can create widgets and drop them in and make a user interface. You can use all the standard user interface libraries. We see a lot of companies who make things like cash registers and airport kiosks. They're using Android to do these things and Android Things provides the same toolkits to make it much easier for them to do this.

[SPONSOR MESSAGE]

[0:40:39.4] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW CONTINUED]

[0:42:15.5] JM: What is the market of other hardware things that you might be connecting to your central Android Things control device? Maybe you can give me an idea for the hardware architecture. Like if I want to build some – it's like a Roomba, but it's also got a cash register on top of it and its location-aware and things like that. I need several different modules, right? I need the mobility, like it can drive around my apartment and vacuum my apartment, but it can also maybe drive up to people that enter my apartment and charge them to –

[0:42:53.2] WP: It sounds almost you're building a restaurant waiter robot. It's like a robot that vacuums the floor while it's taking orders.

[0:42:59.7] JM: I think we've got a startup idea right in front of us. Is there a market of those kinds of modules? Can I take those things off the shelf and build that for a couple hundred bucks?

[0:43:11.5] WP: Well, maybe not for a couple hundred. I mean, I guess the problem is that buying things like the robot vacuum cleaner component, you can't necessarily buy a robot vacuum cleaner. There are companies who make them, but the design of the vacuum cleaner itself is something that only a small number of companies make. In the case of my dog feeder, I actually went to the pet shop and I bought a dog feeder. You can buy these things already that they have a very simple button. They have a timer in it that can time the dog feed.

I took it home, I gutted all the electronics out of it and I put my own electronics in there. Someone did the hard work of building the plastic housing and the motors and all that business. If you wanted to do a proper company, at some point you need to manufacture your own plastic housing and the motors and all the other stuff.

Then what you'd have to do is you would get – you would hire a design firm who has the ability to make plastic injection molded parts, and you would have to make that. Now that's one of the

things about IoT that we haven't addressed yet. In the past, making electronics was very hard and we've tried to reduce the cost of that. Previously, if you wanted to make some electronics for a custom piece of hardware like this, you would have to take a CPU and memory and all that stuff. You would design your own 16-layer board, you would try to check the schematic, you would have it fabricated and it would be very expensive, because making 16-layer boards is hard.

It would come back a couple weeks later. You'd do a bring up, try to test the software, it wouldn't work, you would fix a few things and iterate on that. It could cost many tens of thousands of dollars to build a prototype that way. Now we've helped with the hardware side of things with our system on modules, so that makes that part really easy. What we haven't addressed yet and where I think there's room for innovation is for quick prototyping ability.

People using 3D printers right now, but you can't really use a 3D printer to make a commercial quality device that's robust and strong and that doesn't fall apart, or whatever. There's still some room for innovation in this area to help companies make products, like the physical products that go around. If you want to make a robot vacuum cleaner, it's one thing to write the software and build the algorithms for it, but then you need someone to build the physical plastic robot cash register, vacuum cleaning thing as you proposed.

That's where it gets interesting is the hardware side of things is still a challenge in building the physical prototype. If you're a startup who's building devices, you're going to need to have some manufacturing capability to build all the plastic and the bits and pieces that go around it. However, the software has always been a hard problem. There's actually many companies out there who have the ability to make plastic things. For them, software was the hard part, and so Android things helps to address the software side of it, and they can now focus on their core business which is manufacturing plastic widgets.

That's nice, because if you're a company who makes plastic widgets, you don't want to have to hire your own Linux kernel engineer and your own person who understands OpenSSL and can make patches to AOSP. You don't have to do all that, so you can focus on making your plastic devices, you have one programmer who writes the APK, and you can stick to your core business and basically someone like Google can worry about the security for you.

In terms of designing, you mentioned at the start of your question how would you architect the internet connection and whatever, add devices are Wi-Fi enabled, that comes automatically with Android Things. You might want to use other wireless protocols, such as OpenThread, or something like that for the devices to communicate to each other on a local network. They don't necessarily need to use Wi-Fi. You might have very low-powered devices. You can buy alarm systems that have door switches, where the door switch is powered by a very small battery and it uses OpenThread that every time the door opens, it sends a small packet of data and that then goes to a central collection point where then it talks to the cloud. If you're building an alarm system, that would make the siren go off if the door opened.

Using wireless protocols like OpenThread, you could have battery-powered small devices floating around. In the case of a restaurant, it could be one of those vibrating pages that pages the customer to tell them their table is ready. You might use OpenThread and a very small microcontroller in that device, and then you would use a bigger platform like Android things at the center of it all to collect all the data and upload that data to the internet, because you don't necessarily want every device talking directly to the cloud. That drains a lot of battery power. It might not be necessary. There's many things to think about when you're architecting a solution like that.

[0:47:38.5] JM: Just so I understand, if I wanted to build the robotic vacuum cash register, one way I could do it is hopefully there's a robotic vacuum distributor that's built with Android Things, and then a cash register distributor that's built with Android Things. I put these together on the same hardware device and then I have maybe another Android Things device that's the central Android Things component that talks to the cloud. Is that what I need? Do each of these things need to have an Android things interface to communicate with each other?

[0:48:15.7] WP: Android things is not a network protocol. If three different companies built three different devices with Android things, that doesn't mean they can talk to each other. That would require the three developers to communicate with each other and say, “Hey, let's agree on a cash register protocol, or something like that.”

Android Things is cloud and network agnostic. We basically provide an operating system and we provide an HTTPS stack and you can do whatever you want with it. Now it's up to you as the developer of the product to add a network protocol. Now if you build a dog feeder, you might not other people manipulating the state of the dog feeder. It might cause problems, or it might not be what you're intending to do.

If you wanted to have an interoperability standard, that would be something else. Now Google also has the Google Assistant and we also have an API called smart home. Smart home is our API that allows every IoT device in the world to talk to the Google Assistant. If a device enables smart home, it means you can tell your Google Assistant, “Okay Google, turn on my lights on, or turn the vacuum cleaner on, or something like that.” It will then communicate to all the devices that you own to make them do the action that you want and coordinate it amongst each other. Smart home is the API for that. Android Things itself doesn't specify any network protocol itself.

In the future, say hypothetically a couple years from now there are all these companies making these devices. If they all agreed to work with each other, they could build a coordination protocol to make those things happen. We're still in the early days of IoT, so standardized protocols where every device can talk to every device is a challenge, because of course everyone can't agree on one protocol. This is a common problem with every field in every industry that's ever happened is that there's different standards between countries and companies and whatever else.

These things take time, but gradually we're seeing more standardization, because people are seeing the benefits of being able to buy five different light bulbs from five different companies and actually control them all at the same time, because no one wants to have to bring out five apps on their phone and change the state of each light on each app. It's much easier to say turn on all the lights and every light turns on in a coordinated fashion. That's where the smart home API comes in for the Google Assistant is it helps take care of coordination tasks and things like that.

[0:50:33.7] JM: Sorry to harp on this, but just if I want to integrate different modules together, basically it's going to depend on whatever the interface is that each of those modules is providing, whether it's a temperature sensor, or a GPS sensor, or a module as big as a cash

register, or as big as a robot-connected vacuum cleaner. Presumably, you're buying these things and integrating with them, because they present some developer platform to you and who knows what developer platform that is, but probably you can integrate with it on Android Things. It's just going to depend on the platform.

[0:51:11.0] WP: Okay. Now I understand. There's a lot of components. When a consumer buys things, they come inside plastic boxes and you hook them up with Ethernet. However, when a developer buys things, so if you're a developer and you want to buy a temperature sensor for your Android Things device, typically temperature sensors talk with a protocol called I^2C, or sometimes a serial port. A GPS module for example, typically they communicate over RS-232 serial port protocol.

There are standards for GPS's for example, GPS has a standard called NME I/O 183. That is a serial protocol that says if every GPS speaks this serial strings, any software could read them. You buy your GPS module, it talks NME I/O 183 and you read it over a serial port in your Android Things application. When you plug the temperature sensor in, that's an I^2C device. Android things supports I^2C natively, so you could bring that data in to your Android app as well.

We support USB, like if you have a USB audio input that plugs in as well. When you plug in the robot vacuum cleaner to the robot cash register, cash registers also would implement an RS-232 protocol, where you could send commands to the serial port like, "Hey, open the cash register. Close the cash register. Ring the bell," things like that. The cash register and the robot wouldn't necessarily run Android Things. They would be peripherals that plug into a central Android things board and that would allow you to integrate all those things together.

We support all of the standard device interfacing protocols that people use. I^2C, SPI, RS-232, I2S, GPIO. Yeah. We support all of those standards, because those are the standards that are supported by development modules when you buy these raw components that are integrated together, that's what people are using, and so we support all of those.

[0:52:57.6] JM: I like the fact that you envisioned this cash register actually having a bell, like "Cha-ching."

[0:53:03.7] WP: It's like the old days.

[0:53:04.1] JM: It's a great image. You gave me a lot to think about. This has been really interesting. I want to ask a little bit about a very tangential topic. I was looking at your LinkedIn, I think you have a PhD in augmented reality, is that correct?

[0:53:16.6] WP: Oh, yeah. Yes. From 1999 actually until the year 2006, I used to build outdoor augment reality backpacks. We would go outdoors and we were one of the first people to take a full computer outside that you could wear outdoors. You could actually use gloves on your hands to manipulate 3D objects outdoors. You could create buildings and you could move trees around and you could walk around and visualize 3D objects in real-time outside. We have this sub-centimeter accurate GPS unit, so you could do real-time registration. Yes, I used to do a lot of work in that area many years ago. Yes.

[0:53:49.1] JM: That's really cool. Now we're at this bleeding edge of augmented reality right now, not to say that augmented reality doesn't overlap with Internet of Things, or that I mean, it sounds like your background was just as much of an Internet of Things background as it was an augmented reality background. Why aren't you for example on the AR team? Do you feel AR is maybe, is it a little too early still, or do you – is it arbitrary? What's your sense there?

[0:54:18.0] WP: It's interesting. I've actually worked on many different things in my career. I've done augmented reality. I then worked for a virtual reality startup. I've built IoT devices. I actually used to also build at my university when I worked there, we also did projects with watches, which got me excited when I started at Google because then we were working on Android Wear. Then I got into wearables.

I've moved around between different things over the years. I've always been just generally excited about new technology in general. Whether it's wearables, or augmented reality, or mobile computing, or flight simulation, or building IoT devices, I've done a lot of things because I think these things all come together at some point in the future. I'm just generally very excited about technology and all the cool things you can build with it, so it's about what fun problems can you solve, what things can you do to help people?

There's a lot of people who don't understand technology and also there are people who don't have – there are people with disabilities and things like that, and how could we use technology to make their lives better? Because we have so many interesting and cool devices now that it's sort of, what can we do to help people? There's all kinds of interesting problems to solve. I think for new developers who are getting started, there's lots of things you can do and it's a matter of taking your creativity and going, "What could I build that could change the world and what problems do I have that I could solve?" It's what I've been doing my whole life is just exploring different things and I look at a problem, like how could I use a computer to help fix that, or do something interesting?

[0:55:43.7] JM: Now, I was talking to a somebody else who had spent a lot of time in augmented reality development several weeks ago. One thing he said that surprised me was he thought that augmented reality, at least in the glasses world, the idea that you would have augmented reality glasses, he was like – I don't think it's going to be as big as people think, because people don't want to have their vision occluded. You don't want something that is between me and you on a screen and you don't like this asymmetric information.

He thought that that audio augmentation was going to be a much bigger deal, rather than the visual augmentation. I'm just wondering do you have any – I know we're almost out of time and this is not what you came on the show to talk about, but do you have any predictions, or timelines, or perhaps things I wouldn't hear in other places? Yeah, because clearly you've thought about this a lot. Do you have a vision for how the augmented reality applications are going to unfold?

[0:56:41.8] WP: AR is a funny business, because it's been around for a long time. When I did my work – so I mean, everyone thinks AR just was invented five years ago, but it's actually been around since 1968. There was a professor named Ivan Sutherland who in 1968, he built the world's first virtual reality, augmented reality system. It rendered a cube and it was very simple, but a lot of the principles that were pioneered in that system ended up forming the foundation of OpenGL pipelines and optics that we use in HMDs and so forth.

It's funny, because while computers have improved, like the CPU performance has been improving with Moore's law every year and things are getting smaller and tinier, the actual optics

for head-mounted displays aren't improving at the same speed. It's a really hard problem. People keep talking about, “Oh, we could have AR in a contact lens and we could make it hide and everything.” It's literally with current physics, it's impossible, and it requires a breakthrough of a magnitude that we can't even comprehend to be able to get through that.

If you want a wide immersive – wide field-of-view display, you end up having to wear a large helmet that covers your field of view. You can't have a tiny, tiny display that also has a wide field of view. You have to choose one. You can't have both. It's funny, because over the years when I was working on AR in 2000, I borrowed a lot of concepts from the 60s and 70s and 80s for my work, and a lot of the stuff while the graphics are really cool these days, the actual interaction techniques and the sensors and so forth, they're a little bit better, but they're not a million times better.

I think AR and VR are still a long way out for what people think. If you ask most people like what do you want it to look like? I want a tiny contact lens that I can put in my eye and see everything. Literally, that may be impossible. It might take a hundred years, or it may take even longer, or I don't know.

Things are improving, the graphics are great. Actually, I have an oculus at home, I play with that sometimes too. I mean, they're really fun and the graphics are fantastic, but you have to appreciate the technology for what it is and it's not solving other problems, like haptic feedback and whatever, are still very early days as well.

This concept of people living in these worlds where they can play any game and grab physical tools and manipulate them and do things like that, it isn't going to work as well as people hope it is. There's still a lot of work to be done in AR. I'd encourage everyone to go and look at the work Ivan Sutherland did. The graphics wasn't great, but it's funny how a lot of the optics and the designs of these things are very similar to the way they used to be. Instead of – Yeah, I don't know. It's the system I was building 20 years ago whatever, the graphics are different but the core problems are the same and sensor tracking and being able to walk around large spaces.

A lot of people are like, “Ah, it would be great if we could build an HMD that had no cables.” It’s like, “Sure, but Wi-Fi – sending video signals over Wi-Fi is hard.” It adds latency, it then requires

you to carry a battery on your head. Sometimes having a cable is actually a good thing, because it means you can put the power supply and the computing somewhere else.

Engineering is all about tradeoffs and it's about you have to – you have price performance and size. Choose two. The other one is always bad. It's very complicated. Moore's Law is not going to make HMD super tiny in the next 10 years, or anything like that. The stuff we have now is amazing and I love it.

Technology is fun. I mean, there's so many interesting things to learn and yes, it's very interesting what we've been able to do. If you look back a 100 years ago, electricity didn't exist, airplanes didn't exist, virtual reality didn't exist. Look at all the amazing things we have now and it's exciting as to what might be able to be built in the future.

[1:00:17.4] JM: Couldn't agree more. Well Wayne, I want to thank you for coming on the show. It's been really fun talking to you.

[1:00:21.7] WP: Cool. Well, thanks very much for your time. I hope everyone enjoys it.

[END OF INTERVIEW]

[1:00:27.2] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]