S03E02: Shaping a more sustainable world with SDVs: In conversation with TTTech Auto

Stefan Poledna (00:00):

We have just a finite amount of the materials on our Earth, and we need to be extremely careful in a way how we utilize that. And it would be really a pity and a shame that we just have to dump so much material just because the software cannot be upgraded or we cannot upgrade some hardware.

Kyle Fox (00:33):

This is a Smarter World Podcast focusing on breakthrough technologies that make our connected world better, safer, and more secure. I'm host Kyle Fox. Each episode, we introduce bright minds and their approach to a more sustainable world. We discuss the opportunities and challenges they face and how technology can change the world for the better.

(00:51):

Today, we're entering the complex, or maybe not so complex, world of software-defined vehicles. Today, I'm joined by Dr. Stefan Poledna, CTO at TTTech Auto. TTTech Auto tackles the challenges of future vehicle generations, as they specialize in safe software and hardware platforms for automated driving and beyond. With their technology, they aim for safety and electronic robustness for a more automated world. I'm looking forward to an exciting discussion about the vehicles of the future and how SDVs can help shape a more sustainable world. Welcome, Stefan.

Stefan Poledna (01:27):

Welcome from my side. I'm glad to be on the podcast.

Kyle Fox (01:30):

Before we dive into the deep waters of SDV, please tell our listeners a little bit about yourself and TTTech Auto.

Stefan Poledna (01:35):

I am CTO of TTTech Auto. And so I'm looking into technology and also in new business creation. I have more than 30 years of background in the automotive industry. I've been working there, basically, for my whole life. I have also co-authored several patents. Actually, it's more than 80 patents. I received a Master's and PhD degree at Vienna University of

Technology. Also, lectured there. So that's a little bit about my background. Other than my professional background, I like to be in the mountains.

(02:06):

A little bit about TTTech Auto, so we spun off TTTech Auto five years ago, because we developed the first safety domain controller for automated driving back then. And then we saw that this trend for a more centralized domain computing really took off, so we spun out TTTech Auto as a separate company out of the TTTech Group. And there, we focused on safe software, on hardware platforms, on software platforms, and driving forward this complex integration topic to bring integrated highly complex system to the market.

(02:47):

And we are about 1,100 employees in the company. We have around about a thousand software engineers, so that's really the focus of the company. In 2022, we raised 250 million euro with Aptiv and Audi in our last funding round. And we use basically these proceeds, and we are moving forward now in the SDV, the software defined vehicle, space to really drive forward this aspect of safe computing.

Kyle Fox (03:19):

Fantastic. With your background, 30 years in auto, as you said, spending your entire career on automotive, and also your educational background, you have the experience to drive these sorts of things. And TTTech Auto, it sounds like, significant funding coming in, and y'all have been at the forefront of this topic. The automotive industry has been talking about software-defined vehicles for years now, and there are many different definitions for this term out there. So maybe as we unpack our journey here, let's start with, how do you define a software-defined vehicle? And why is this concept such an essential step toward a more sustainable world?

Stefan Poledna (03:51):

Okay. Trying to unpack that, I'd like to take, first of all, a customer perspective. So you need always to think, what's the benefit from the customer point of view? And with software defined vehicles, customer perspective needs to be that customers get, let's say, relevant feature upgrades, new functionalities via software during the lifetime of a vehicle. So it enables you to get something new, better during the lifetime of the car.

Kyle Fox (04:23):

So you're talking about instead of having my same infotainment system for 10 years and it becomes really out of date, you're actually talking about upgrading that particular system and other systems, correct?

Stefan Poledna (04:33):

Yes. And let's say we want to upgrade that from a software perspective, so like your smartphone, you can get new apps. After many years, the hardware becomes outdated. That's clear. The hardware has a certain run rate and certain life expectation. But looking at your phone, there's so much you can do. If a great new app comes up, you can simply download the application on your phone, and you have a new feature on your phone that wasn't there before. So that's the same story on the car.

(05:04):

And if you want to make that a reality, you need a couple of ingredients to do that. First of all, you need to have enough compute and communication resources to do that. If you have a system in a very, let's say, traditional architecture where you have a hundred tiny little ECUs each connected to each other in some way, then you don't have the headroom to bring in new functionality, because there's simply no single space where you can do that. So it means you need to go to a more centralized compute, you need to have much more memory, more compute performance in order to have headroom to bring these functionalities. So that's one aspect.

(05:47):

You had, also in your question, this aspect of more sustainable. So if you have a car and you have a lifetime expectation of, let's say, 10 or 11 years or something like that, and you can refresh the car constantly in terms of software upgrades or even you could imagine you have a blade or a board upgrade even to refresh some of the compute hardware as well, then the majority of the car that is really material energy intensive, you have one and a half tons or two tons of car, that's a lot of material. There's a lot of manufacturing, there's a lot of energy going into that. And if you can keep that fresh and up-to-date just by software upgrades or by tiny pieces of hardware upgrades, this is a very attractive solution to not have to buy a car after, let's say, five or six years because it's simply outdated in terms of functionality.

Kyle Fox (06:40):

I totally get it, and I loved how you showed the analogy of using a phone. Some new exciting app comes out, and even with my phone, which is a couple years out of date, I have a really good chance of being able to run that. But I think you're also extending this beyond just infotainment, like I'm playing a new version of some social media app. You're actually talking about potentially upgrading the guts of the car itself. Maybe a more efficient braking system.

Stefan Poledna (07:04):

Yeah. And that then goes really to the core of Metas. Because upgrading IV functions, that's the easy part. And the tricky and the hard part is, if it goes really to the core of the functionality of the car, if it goes into the safety domain, if it goes into security domain, if it goes into chassis, if it goes into driver assistance functions, automated driving, then it really gets tricky. And that's the reason why we coined this term, which we call 4SDV. So we say it's not a single S, it's 4S in SDV that we have to consider. It's the system's approach, it's safety, it's security, and it's software defined. So it's 4S.

(07:52):

Because if you take this perspective and you look at automated driving, improved powertrain performance, all of this, there's clearly a safety side of that. There's clearly a security side of that. If you take automated driving, it needs to stay safe, if you do a software upgrade. You cannot just simply put features in and release every week something that seems cool. You need to do all the validation. And if you look at safety, if you look at security, then these are clearly systems properties. And system properties means it's a conjunction of the overall architecture of software and hardware. So if you have a safety target, for example, let's assume you have an ADAS system and you have automated emergency braking. If somebody cuts in front of your car, a child runs in front of your car, then you need to guarantee that the braking action takes place in a couple of hundred milliseconds. So it needs to be really, really quick, and that's about safety. You need to guarantee that.

Kyle Fox (08:54):

I would imagine that if you've got... Just the security side of it itself, being able to push that software into that vehicle, even if it is honoring the safety side of it, you got to make sure it's done securely. And the challenge is there, because you don't want the newest, latest thing to cause you to not be able to avoid that child on the road, as you said.

Stefan Poledna (09:12):

Yeah, absolutely. And on the security side of things, you come to hardware, you need a root of trust, you need encryption, you need everything that enables you to make sure that the software update only gets in your car if it's authorized, if it has been approved, if it's good, and nobody else can take over control of the car.

Kyle Fox (09:34):

You'd mentioned the 4SDV approach that TTTech Auto is pioneering. How is it for tackling these limitations that you're describing?

Stefan Poledna (09:42):

I'd like to bring in a second concept that's very important. What you do in essence with software defined vehicles, you try to extract the hardware part from the software part, because you need to bring in new software versions on top of an existing hardware. This requires a certain separation of the hardware capabilities, and then you link the software capabilities to your hardware capabilities. So this hardware extraction needs to be managed by a platform, and the platform needs to take care of all these basic functionalities. So if you upgrade some software, whether it's safe and secure, it needs to manage, as it's not only a single function that runs in the car, it's multiple functions running at the same time.

(10:27):

In the beginning, I said, you need to have headroom, you need to have enough compute power, you need to have enough memory. So this directs to very strong domain compute and ultimately to more central compute architectures. Because there, you have the high integration, there, you have the very high performance SOCs. And there, you can host your complex software functionality. And it's a multitude of functions that are hosted there, so you need to have a strong platform that gives you the safety, the security guarantees, and that manages basically all the resources in terms of compute, memory, communication, and to make sure that those activities that are really critical get enough resources to manage their real-time safety and security targets.

Kyle Fox (11:19):

You're describing taking multiple discrete components and pulling them together into essential compute. That starts making me think that you're going to have to look at a lot of techniques to guarantee that real-time communication coming in, and then also being able to have the hardware actually be, in a sense, virtualized so that it's able to provide different resources to different software loads in real-time. Is that correct?

Stefan Poledna (11:46):

That's absolutely correct. And you're really pointing to the core of the matter, because it's about managing these resources. And if you come back to this example that I said with the automated emergency breaking, if you dive in a little bit deeper into that, then what has to happen there? You have a multitude of sensors, you may have radar, you have multiple cameras, you might even have a laser.

Kyle Fox (12:09):

Right.

Stefan Poledna (12:10):

So you need to have sensor pre-processing. Then you have some communication going to a compute unit, there, you do sensor fusion. You need to bring all these signals together to establish a map, what's your driving direction? What's the objects around you? What is the category of these objects? What's the trajectory, the moving speed? Based on that, if you have the sensor fusion, you need to plan your trajectory. So what does it mean for you? Can you avoid? Do you need to brake? Do you need to steer? So you have to plan a trajectory. This could be to go to the powertrain, to the brakes, the steering. Might even go to chassis, function.

(12:50):

And all of this complex processing flow needs to be managed in a small number of a hundred milliseconds. So there, you need to manage all the resources, yeah, you have a lot of compute, you have a lot of communication. You will have, for the image processing, special GPUs or deep neural network accelerators. So all of that needs to be managed in terms of resources and communication, and that's something... When you have more of this function going in parallel, this is typically what you do in a platform, and the platform gives you, you can call it virtualization or you can call it partitioning, so that you make sure you have different functionalities and each of those functions has enough resources to do the job in order to meet the real-time and safety targets.

Kyle Fox (13:41):

And that's where TTTech is innovating with the 4SDV approach, right? You're going to have some established communication protocols within the software itself saying that, "This is how I'm going to use the hardware," and being able to share information back and forth. And that's really where y'all are innovating, correct?

Stefan Poledna (13:57):

Yeah. Absolutely. The classic approach was typically to set priorities and say, "Okay, this is important. This is more priority than that." And then you try to tune the priorities that overall, your response times, your behavior, resource management is okay. If you have so many compute cores and if you have so many communication links, there's CAN, there's Ethernet and all of that in parallel, then it's very hard to fine tune the system that you can really give a safety guarantee from multiple parallel running applications. And that's really the challenge.

(14:33):

If everything goes well and the system is lightly loaded, usually all is fine. But you have the rare case scenarios, the corner cases, where it really can get wrong. And that's almost no longer testable, because if you have so much functionality integrated together, then the testing becomes more and more complex and ultimately impossible.

(14:55):

So we can synchronize all the communication, and we can synchronize the CPU cores, that they share a common time base. And based on the common time base, we allocate functionality, who is doing what at which point in time. For example, if you have the sequence that I told you a little bit earlier from the automated emergency braking, then you can allocate, "Okay, I have the pre-processing, and I start here with the pre-processing. Then there's the communication going to the central unit that builds the sensor fusion. Then after the sensor fusion, I'm going to do that and I'm going to do this." And I can really align communication and processing activities along a timeline.

Kyle Fox (15:39):

Based on time.

Stefan Poledna (15:41):

And I can monitor that everything is taking place within its deadlines. So I can monitor the system on one hand side and really make sure that in a very distributed communication processing scenario, everything fits fine.

Kyle Fox (15:58):

That's brilliant. Because if you have a tree of a thousand priorities that you all have to go and test and do path fine testing, the matrix of test cases in there, like you said, it approaches, I don't want to say infinity, but it's certainly unsolvable. And what you're turning is on its head. You're saying, "We're going to do this over a timeline, and I'm going to give you a job and you a job, and all you have to do is make sure that you honor that timeline and you can go and do your job correctly." So it allows you to break the paradigm completely.

Stefan Poledna (16:26):

And what we do is we shift now from testing, and you said it's close to infinity, and I have a fun example for you on that. We have worked for 20 years on some clever tooling that uses heuristics so that, in this haystack of possible settings, we find the right way that we have the resources allocated in a way that it fits. And we do this upfront with a computer, and we have various smart heuristics, and we have a lot of PhDs working on this topic and constantly improving on that. So we do it once, and once we have found a solution, we know it's a valid solution, that we only use those valid solutions.

Kyle Fox (17:07):

Interesting.

Stefan Poledna (17:08):

And I can give you a real world example. This car is already driving on the streets. It's a system for automated driving and automated parking. It has close to 50 CPU cores, it has five hardware accelerators, it has two TSN switches with gigabit speeds, and it has around about a hundred different functionalities are integrated, so that a lot of lane keeping and warning and all this stuff. Some of that is really safety critically, more is more informative.

Kyle Fox (17:38):

And this is a vehicle on the road today?

Stefan Poledna (17:40):

Yeah, it's on the road today, that vehicle. And if you do the calculation on the numbers, so all the possible allocation of tasks, these compute resources and communication would be 10 to the power of 80.

Kyle Fox (17:54):

10 to the power of 80?

Stefan Poledna (17:56):

Yeah. And that's more atoms than we do have in our known universe.

Kyle Fox (18:03):

Wow. That is as close to infinity as you can possibly describe.

Stefan Poledna (18:06):

Yeah. So even all the atoms in the universe would not be sufficient to enumerate these possible variations. It's simply a huge number. Yeah?

Kyle Fox (18:15):

I love it. Just like mathematicians and physicists, when they encounter infinities, they get rid of it. That's exactly what y'all did. You said, "Okay, this is an infinity. We have to get rid of it."

Stefan Poledna (18:24):

Yeah. And we have approximately... Out of these 10 to the 80, we have 10 to the five possible solutions.

Kyle Fox (18:30):

Still a big number.

Stefan Poledna (18:32):

Yeah. Still a big number, but we need just to find one.

Kyle Fox (18:36):

That's the brilliance of it, right? It's almost like NP complete problems. If we could solve one of them, we'd solve all of them, right?

Stefan Poledna (18:41):

Yeah. It's an NP complete problem. It's absolutely an NP complete problem. You need to have heuristics, you need to have a smart approach to searching. But the big advantage is you do that when you design your program and you don't have to test for ages to get at least a certain level of confidence that you have not missed something.

Kyle Fox (19:03):

Wow. You got my mind just churning on this. And just for our listeners' benefit, I'm sure they're following along, but I think there's two things that have come out of here. One is we want to be able to upgrade these vehicles. After five years, six years, you might be getting a little bit aged in what the vehicle can do. And it's no longer getting aged, because I can't play the latest version of Angry Birds, my simple example. But your braking system, or your avoidance algorithms, or the neural nets running to be able to do some of your ADAS may have a massive improvement. And you want to be able to bring that in rather than buying a new car. So there's a huge sustainability side of this.

(19:42):

But you guys are tackling this from the other angle as well, and just our previous discussion is that if you were to try to enumerate all the use cases and all the different permutations in a classic way of tree problem solving, you literally could spend the entire age of the universe trying to test the thing to get it to work. So you're coming at it from a completely different angle, saying, "We're going to do this, and we're going to find a solution that works and then be able to apply it to all of them." But something that is well over more orders of magnitude more efficient than the classic way of doing it, which isn't going to work in the first place.

Stefan Poledna (20:16):

Yeah.

Kyle Fox (20:16):

Did I get that right?

Stefan Poledna (20:18):

That's absolutely right. Yeah.

Kyle Fox (20:18):

Fantastic.

Stefan Poledna (20:19):

Deep to the heart of the SDV. Because if you want to bring software upgrades, then you need to control the amount of testing and integration and validation efforts that you have for each of them to bring. Because if we say that the promise is to keep the car innovative and to keep some updated features in the car, then on the safety side of things, it's really about validation and making sure that your integration is robust and it works and it keeps all the safety properties. So if you bring more and more functions together in a centralized compute unit, then this validation and integration problem becomes even more complex. So this is what you need to address to make the SDV a reality and to make it really a success.

Kyle Fox (21:10):

Wow. This is a lot to go through, but what you're describing is we're approaching a brick wall if we don't do something different. And that's exactly what y'all guys are doing-

Stefan Poledna (21:17):

Yeah.

Kyle Fox (21:17):

... is taking it this completely differently. We were heading toward a insolvable problem. And if you think about it as a parent, if I'm going to buy a new car, I care about how safe it is. And I'm part of the generation that couldn't upgrade their cars. Maybe you could put some aftermarket stuff in there. But it sounds like the buying habits of people are going to change if they have the opportunity to say, "Hey, we're going to buy this car, and three years from now, there is going to be an upgrade or the chance of an upgrade and to make it more safe." I don't think... That's never existed, right? There's never been a five years later, your car is more safe opportunity. And that's exactly what you guys are trying to build.

Stefan Poledna (21:54):

Yeah. That's the idea, that you can improve there. Yeah.

Kyle Fox (21:58):

What are some of the barriers here in terms of shifting to a 4SDV approach that y'all are encountering?

Stefan Poledna (22:05):

I think it's basically two steps. The first step is you need to bring headroom into the architecture. If you don't have headroom, you cannot upgrade. It's very clear. If you have a very segmented architecture, no headroom, no spare compute cycles, no spare memory, you don't have a chance. So you need to go for more centralized architectures, high performance, and you need to keep spare processing, communication, memory capacities.

(22:33):

And the second step is really thinking about how to integrate the software, how to release the software, how to validate the software, how to make sure that this release cycle is really going well. And from an OEM perspective, so from a car manufacturers perspective, it's multiple parties. You have somebody that is responsible to build the hardware side of things, then you have multiple software partners for diverse functionalities. You have SOC partners that bring in the compute chips, which are at the core and the heart of this. So all of these need to play together, and you need to have a complete release pipeline that you can manage that with all the partners included. So this goes into this topic of CI/CX software factory, build chain, build automation.

(23:23):

And I think the third aspect that was the one we have been discussing before, and this is why we are coming in with the 4SDV approach is on the safety side and security side of things, you must not be naive what the challenge is that you're up against. Because all of a sudden, you had one ECU that was doing the breaking, you had one little ECU that was giving you the surround view, and another one for this and that. Now, all of a sudden, it comes together in a single central compute, and now, the integration challenge becomes a total different animal.

Kyle Fox (23:57):

The benefit of having these smaller ECUs, you could say it's basically a closed box. It does its one or two things, and it's done.

Stefan Poledna (24:04):

Yeah.

Kyle Fox (24:05):

And now, you're sharing it across some of these processors.

Stefan Poledna (24:07):

Yeah. So you had a very, let's say, natural, very fine-grained partitioning in that concept. It gave you a lot of these partitioning safety properties and things like that, but at the same time, you couldn't upgrade.

Kyle Fox (24:23):

Talking a little bit about the processors that are actually in the car, would you see a car that literally has dark silicon inside of it? Dark silicon, meaning that you have it set for version 1.0 with a certain amount of processing capabilities and literally have a processor next to it that is for future upgrades that actually isn't used at the beginning?

Stefan Poledna (24:42):

Whether it's that concept or it's simply you over-provision that you say, "Okay, I put in, let's say, 16 core version that I just need eight cores, and I put in twice the amount of memory." That's also an option that you can do. Or you over-provisioning the communication links and say, "Okay, I do 2.5 gigabit link, and I only load it with 200 megabytes." All of that is happening.

(25:09):

And when you're bringing up the dark silicon, one option there is you bring in that, but you don't bring it for the full fleet. You bring it in, let's say, of a percentage of the fleet, and you use this percentage of the fleet for shadowing. The idea of that is that you have a small part of the fleet where you can download and process a new version of software in parallel to the old version of the software. And you do an AB testing. So it means you compare how the new version is performing against the old version really on the road.

Kyle Fox (25:44):

Oh, wow.

Stefan Poledna (25:44):

So the new software version is not in control, but you can test it back to the old version and can get data back to validate whether it's really behaving better.

Kyle Fox (25:56):

Oh, that's brilliant. You have a co-pilot there that can't change anything, but it acts as if it is, and you're going to get real-time feedback. I never even considered that. That way, you can get real-time feedback on your new software on the road without actually endangering anybody.

Stefan Poledna (26:11):

Yeah. And that concept is often called shadowing. So you wouldn't spend the pricing performance in every car. But if you have a certain percentage, then it gives you the ability to try it really out without harming anybody.

Kyle Fox (26:26):

And then your 4SDV approach where you're saying, "Look, we need to be able to allocate resources safely and securely," this ability emerges from that. Because in the picture in my head is you're able to take this new load and move it over here to the side, and because you have the infrastructure to say, "This is going to take these cycles over here and it is secure and it's safe and nobody can touch it," let alone it can't accidentally touch the drivetrain, for instance, of the car-

Stefan Poledna (26:52):

Yeah.

Kyle Fox (26:52):

... it actually emerges from the approach that you guys are taking.

Stefan Poledna (26:55):

So if you can do this partitioning of resources being it compute cycles, being it communication bandwidth, then this really helps, because you need somehow to guarantee that.

Kyle Fox (27:05):

This is such a fascinating topic. In the half hour that we've had together, I get the sense that we could have a whole series of podcasts, just to dive into this. At the very beginning, I was talking about, how can software-defined vehicles help with a more sustainable auto industry? And sustainability really comes down to, I think you'd said there's a ton and a half of metal in these things and the amount of water and energy and all the things that take to build a car-

Stefan Poledna (27:31):

Yeah.

Kyle Fox (27:31):

... you don't want to have to sell it out five years from now. It'd be great if you could keep upgrading it instead of being 15 years, 30 years. And maybe that's stretching it a little bit. But that's the main tie here to sustainability, correct?

Stefan Poledna (27:41):

Yeah, it's a very important angle. We have just a finite amount of the materials on our Earth, and we need to be extremely careful in a way how we utilize that. And it would be really a pity and a shame that we just have to dump so much material just because the software cannot be upgraded or we cannot upgrade some hardware. So that's, I would say, irresponsible in terms of how we use and allocate our resources.

(28:09):

And the second aspect is clearly also to be seen in all the big congested cities to go more for automated shuttle services or the cars just used very small fraction of a time. They have one and a half ton or two ton for a single 80 kilo person to be carried around, and then they sit idle in the streets of the cities. If we had smart infrastructure and could have shuttle services where we share those things, it could be much more sustainable than what we have right now. And that's also software defined in a way, because the Uber type of application, getting a ride from A to B and where you can access and you could pull that with others. So why not?

Kyle Fox (28:55):

And I loved how you said that, it is an awful waste if we can't use the vehicle just because of some software.

Stefan Poledna (29:00):

Yeah.

Kyle Fox (29:00):

And I'm not trying to dismiss the complexity that goes with software, but that is an awful shame and an awful waste.

(29:06):

Well, one of the traditions we have here on the podcast is, at the end of every episode, we always ask our guests the same question, how do you, through your eyes, envision a greener world 50 years from now?

Stefan Poledna (29:18):

If I try to answer that question top down, then I think it's basically we have to go from first principles, and first principles for me is we have a finite amount of materials on earth, raw materials, and we have in a way a finite amount of energy on our earth. All the carbon sources, we are running out. They're not infinite there.

Kyle Fox (29:42):

For sure.

Stefan Poledna (29:43):

So oil, gas, it's, for sure, not infinite, so we need to be careful with that resources. And if I'm looking 50 years ahead, then it's really about the question, how can we use these resources in an efficient way then what we do right now? And that includes finding solutions that we can cope with less energy and recycling the materials much more efficiently than what we do today.

Kyle Fox (30:11):

I completely agree with you. It's being able to do more with less. That's certainly moving us into something that keeps our world greener and cleaner and doesn't poison the atmosphere. All the points that we've talked about today.

Stefan Poledna (30:23):

Yeah. And I think maybe as the last word, the nice thing about it is the computer is the most efficient machine that we have ever built to optimize things. It's totally universal. A computer is the most universal tool to optimize things. If you use it the right way, it holds really a lot of promise.

Kyle Fox (30:44):

I never really consider that, but you're right. It is universal, you can apply just about any task to it. And compared to other technologies, it actually consumes relatively small amounts of energy and materials.

(30:56):

This has been one of the most impactful podcasts that I've hosted, and there's such a density of information here that we want to unpack, and I want to thank you for joining us today, and I absolutely encourage our listeners to go and spend some time up on the TTTech Auto site. There's tons of material there. Dr. Poledna has done lots of different talks. And Dr., I would love to invite you maybe another year from now to have you come back on the show and we can dive a little bit deeper into how things have progressed over a year. Because I think being able to come back and see what your world is evolving to will be quite fascinating.

Stefan Poledna (31:30):

Yes. Thanks a lot from my side. I enjoyed this discussion very much. Thanks for your insightful questions and a great discussion.

Kyle Fox (31:37):

Thanks for tuning in, and we'll see you on the next podcast.