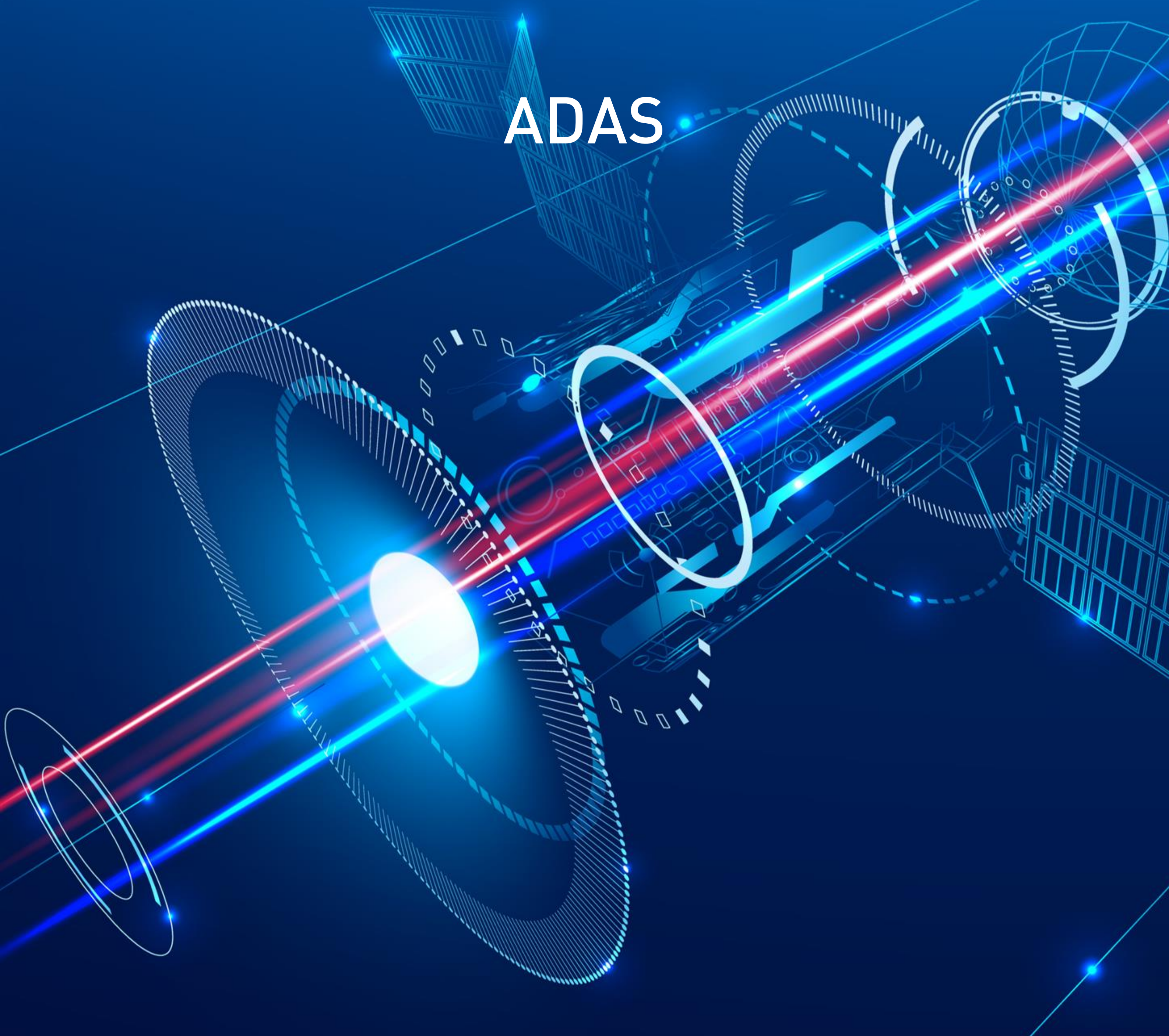
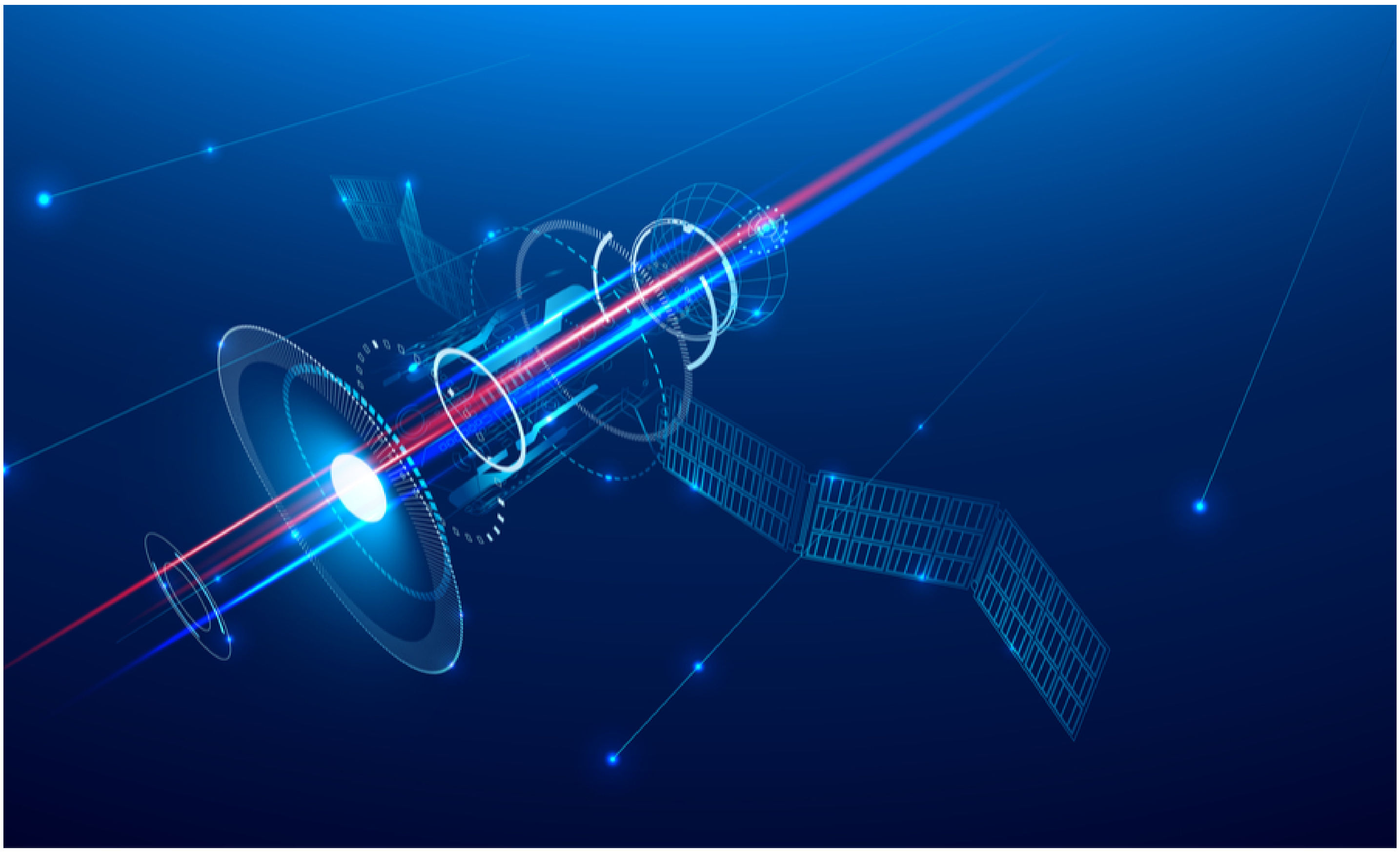
ADAS





ADAS

The automotive market has been growing and is promised to grow even more. The growing demand for ADAS features and other in vehicle systems and features has made electronics the most valuable part of a vehicle. Learn more about ADAS features, augmented reality in cars, as well as in depth technical considerations regarding ADAS software development and ADAS vehicle safety.

Not 100% convinced? Join us as we discuss a variety of topics to help you decide if ADAS is for you, including:

- Growing ADAS Market Will Require Efficient Software and MPU Expertise
- Augmented Reality in Cars: Pros and Cons of ADAS Heads up Displays
- Which ADAS Features Are Most Likely to Lower Premiums and Increase Public Safety?
- Inline Functions in C Language: Why and When They Are Used In ADAS Software
- Neural Network Supervision and ADAS Vehicle Safety

GROWING ADAS MARKET WILL REQUIRE EFFICIENT SOFTWARE AND MPU EXPERTISE

Transportation has a major effect on people and societies, and cars changed life around the world forever. We are now on the brink of another major transportation breakthrough, self-driving cars. While we are still years away from fully autonomous vehicles, advanced driver assistance systems (ADAS) are becoming more common in the automotive industry. The ADAS market is headed for major growth, and smart software developers can grow with them. Developers should be prepared to take advantage of this new era of transportation by making their code more efficient, and mastering the use of memory protection units (MPUs).

My great grandmother grew up on a farm in the middle of nowhere in Kansas (that's actually most of Kansas). She lived until she was 96 and saw a lot of things change in the world. When I was a child I was surprised to learn that she saw Americans go from riding horse-drawn carriages, to driving cars. Transportation has a major effect on people and societies, and cars changed lives around the world. We are now on the brink of another major transportation breakthrough, self-driving cars. While we are still years away from fully autonomous vehicles, advanced driver assistance systems (ADAS) are becoming more common in the automotive industry. The ADAS market is headed for major growth, and smart software developers can grow with them. Developers should be prepared to take advantage of this new era of transportation by making their code more efficient, and mastering the use of memory protection units (MPUs).

ADAS will let us all kick back and relax.

ADAS ACCELERATION

While some drivers currently enjoyed the advantages of ADAS enabled cars, most are unaware of ADAS. That is going to change in the near future, as the US and EU will begin requiring new cars to have automatic braking systems and collision warning systems by 2020. These mandates contribute to the rising ADAS market, which is expected to grow from $22.69 billion in 2015 to $78.19 billion by 2020.

The ADAS market is expected to continue growing as manufacturers push the boundaries of autonomous navigation. Currently, “self-driving” cars are only up to “level 3” of autonomy. Multiple manufacturers, though, are looking towards levels 4 and 5, which will require increasingly complex ADAS systems. Even some companies from unrelated industries are getting involved in the ADAS market. NVidia has announced that they plan to enter the market with Audi and release a level 4 car by 2018.

As more competitors enter the market, software developers will have to work harder to differentiate their programs. Two techniques to set yourself apart are making code that runs more efficiently and mastering the intricacies of MPUs.

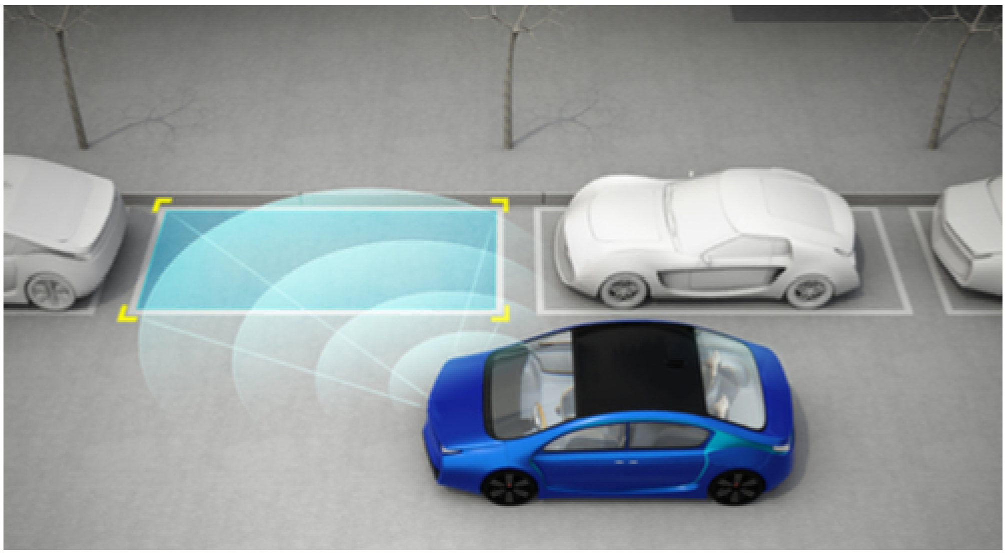
Rear vision cameras will also increase demand for ADAS.

EFFICIENT CODE

One major roadblock on the path to full autonomy is processing power. The level 4 and 5 systems you'll be controlling will need processors that are many times faster than those currently available for automotive applications. You're a humble software developer, not a chip manufacturer, so what can you do? Making your programs more efficient will help reduce the load on processors.

Efficient coding will help with timing safety concerns as well. You don't want to find out that your safety features fail critical requirements after you've already built your program. If you focus on efficiency from the beginning, you're more likely to pass in a timely manner. Libraries can make a major difference in the detection-reaction chain of your program. Using a well-developed standard library can help you parse incoming sensor data much more quickly, making your code run that much more efficiently.

One non-software solution to the processing problem is the move from distributed electronic control units (ECUs) to integrated multi-core microprocessor control units (MCUs). However, the move to MCUs has given rise to another problem, dealing with MPUs.


ADAS will make parallel parking a lost art.

MPU EXPERTISE

While the move to MCUs has helped solve the speed shortage it introduced safety concerns in device memory. Now, with all systems sharing the same processor and memory, cascading faults could cause serious problems. In order to avoid having to certify every system to ASIL level 5, the MPU was developed. The MPU is a difficult beast to master and will require special tools and knowledge to tame it. You should get started on mastering it, because it's not going anywhere anytime soon.

The first thing you should do is to get a good static analyzer. A well made static analysis tool will save you a lot of time in debugging the code for your MPU. In addition, it will allow you to use your compiler to its full potential.

With processing requirements expected to increase as we move forward toward truly autonomous vehicles, the MPU is here to stay. As sensor complexity and variety increases, new cars will require centralized MCU's capable of higher level programming. Those MCUs will certainly require an MPU, which will likely be more complex than the ones we're using now. It's a good idea to go ahead, buckle down, and delve into MPU programming so you're ready for the future of self-driving cars.

DEVELOPER TOOLKITS

As autonomous vehicles become increasingly more advanced, you'll need cutting edge tools to help you ride the wave of the future. The programs governing ADAS systems will need to be fast, efficient, and accurate. This means that you will need the best compiler and libraries available. You'll also need good safety tools to deal with MPUs quickly and correctly. TASKINGhas a wide range of tools available for software developers ready to move into the next era of transportation. These tools will help you write the fastest, most efficient code you can. TASKING's tools will help you speed up your development along with your code with tools like standalone debuggers. Equip yourself with the best tools available, so you can build our future.

Have more questions about program efficiency and MPUs? Call an expert at TASKING.

AUGMENTED REALITY IN CARS: PROS AND CONS OF ADAS HEADS UP DISPLAYS

Do you ever feel a little distracted by all the technology surrounding you? I know I do. When I'm at home I have my desktop computer, tablet, and smartphone all vying for my attention. On the road, I have fewer things to distract me, but the consequences of my mind wandering are more severe. In the space it takes to send a quick text, swipe to the next turn on my maps, or change to a song on a different album, disaster can strike. Advanced driver assistance systems (ADAS) are combating technological distractions by enabling your car to keep an eye on the road when yours are wandering. Another method for mitigating distractions is heads up displays (HUDs) that display information on the windshield. This allows the driver to keep their eyes on the road, even while checking their phone. As cars become more advanced, ADAS and HUDs will start to mesh in an attempt to bring augmented reality (AR) to cars. HUDs a few years from now will display information coming in from ADAS sensors and other more general information. This new kind of HUD will have a variety of big advantages, but will also bring a unique set of challenges that we need to start considering today.

ADVANTAGES OF ADAS HUDS

Current HUDs are little more than a tiny screen that can display things like navigation, messages, or the weather. They came about to help combat distracted driving, which is responsible for 1 out of every 4 wrecks in the US. Some of them just link to your phone and display whatever is on there. Others are a little more advanced and have their own GPS chips, and can even recognize hand gestures. Regardless of what extra bells and whistles they have, all of these HUDs are fairly low level. None of them are able to link to a car's advanced features and incorporate them into the display.

Augmented reality heads-up displays could warn you about what's ahead.

In the near future, as more ADAS enabled cars hit the road, HUDs will need to change. They'll need to interface with ADAS and help the driver respond to incoming information. See, the problem with ADAS is how to display incoming data. When a car's LiDAR system detects something on the road, how does it tell you exactly where that something is and how to avoid it? It can vibrate your seat or blink a light, but a visual cue could be more helpful. If you had a HUD that was connected to the LiDAR, it could highlight the object in the road, bringing it to your attention and helping you avoid it. Some of the current HUD developers are already moving in this direction.

In addition to relaying ADAS sensor data, future HUDs will also need to show information coming in from other cars, infrastructure, etc. Car manufacturers are already working on vehicle to everything (V2X) communication, in order to help give the driver eyes and ears outside their own car. HUDs could feed these data streams directly to the driver. For example, if the driver is behind a tall truck, they can't see the road ahead. A HUD could tap into that truck's front camera stream, and display it on the windshield, letting the driver essentially see through the truck. This kind of system could also display traffic information in the driver's path and weather forecasts as well. Huge amounts of useful information would be at a driver's fingertips, without them having to use said fingers at all.

Even if you've been distracted while reading this article, you can see how this kind of HUD would be useful. Not only would it keep drivers from looking at their phones, it could also help them look out for trouble. However, there are two sides to every coin, and AR

HUDs are no exception.

DISADVANTAGES OF HIGH-LEVEL HUDS

Critics of these kinds of these kinds of systems have a simple argument. Instead of decreasing driver distraction, HUDs add to the problem. If text messages, playlists, or even videos started popping up on my windshield, I think my attention would be diverted. Even life-saving information could become dangerous, depending on how it's presented. If a HUD warning pops up in front of me and obscures my vision, it could cause me to hit something I might have otherwise avoided. Being inundated with alerts from my phone, the cars, and even the road would also be quite distracting. These problems, though, have more to do with how this technology is presented, rather than with the device itself.

Safety alarms could startle and distract drivers.

There are two inherent problems to consider when thinking about AR HUDs: cost and power. These are the same issues that are already plaguing the ADAS market. Manufacturers would love to produce cars with LIDAR sensors that can accurately see in a large 360-degree circle. However, those sensors currently cost around $70,000. One of the current top HUDs sells for $799. Obviously, there's a disparity there, but as cars become more advanced, they'll become more expensive. Every little bit counts.

In addition to monetary cost, there will be processing and electricity costs as well. If automakers start incorporating HUDs into their cars, they'll need to have a processor that can perform all ADAS functions, and then display them. There's also the cost of energy to think of. The powerful processors needed for next generation cars already take up a lot of juice. A HUD will only add to that burden.

These problems are certainly not insurmountable, but they do need to be addressed. If a HUD distracts the driver by sending them safety alerts, it's not just useless but downright dangerous. Consumers will also need to be prepared to pay more for cars with AR HUDs. Manufacturers will need to think about whether they can spare the processing power and battery life to power these kinds of systems.

The other side of the equation is the software required to run a HUD. That's where developers like you come in. Making programs for cars is already quite difficult, and AR HUDs will only add to the confusion. That's why you need tools, like the ones TASKING makes, to help you do your job with a minimal headache. TASKING has developer products such as standalone debuggers and static analysis tools to help you do your job.

Have more questions about HUDs? Call an expert at TASKING.

WHICH ADAS FEATURES ARE MOST LIKELY TO LOWER PREMIUMS AND INCREASE PUBLIC SAFETY?

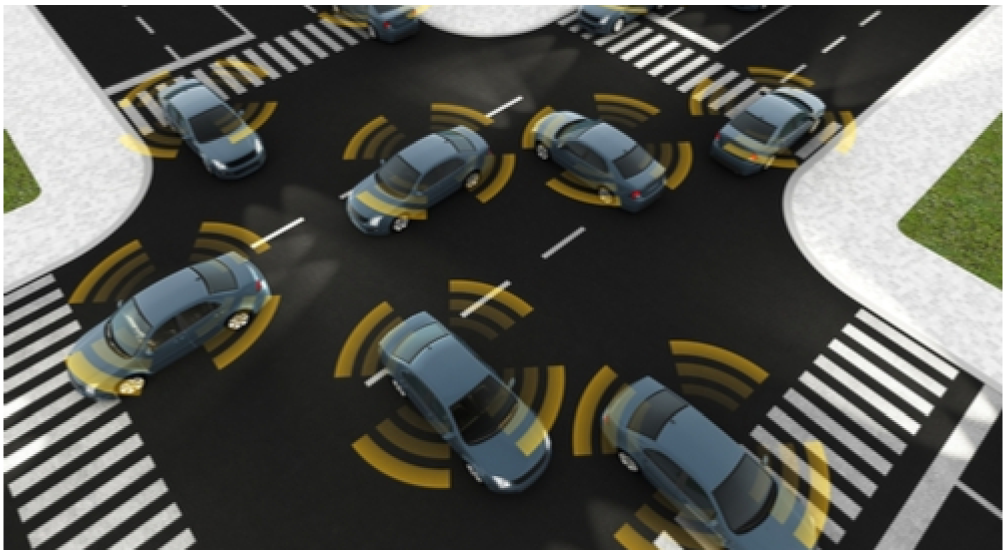
In the near future, we could see fewer traffic jams, lowered emissions, and better overall road safety. But these benefits won't necessarily come from any new laws or regulations. Instead, these dramatic changes might come from our own automotive industries, and the exciting new technologies being developed by the best minds in the field. ADAS functions have the power to transform the way we drive and could influence our entire transportation system for decades to come.

As we see more advances in ADAS features, including advanced sensor technology, real-time traffic updates, and improved mechanical diagnostics, our roads will become safer. This, in turn, will lead to a number of economic and social benefits. In this sense, the automotive industry might just change the world all over again.

ADAS IMPROVES ROAD SAFETY

The number one goal for anyone in our industry is road safety, and ADAS is already revolutionizing the way we think about this important issue. ADAS uses vision sensors, radar, and ultrasound to monitor other vehicles, their speeds, pedestrians, stationary objects, and sudden changes in road quality (such as ice, rain, or gravel).

When it comes to road safety, every second counts, which is why it's so important to build this full safety system from the beginning. In many instances, these ADAS features catch and resolve potential safety threats before the driver is aware of them. As our coding becomes even more efficient, and blisteringly fast 5G speeds come into play, our industry could even make traffic accidents a thing of the past.



5G will help reduce traffic

BETTER SAFETY MEANS LESS TRAFFIC

Our industry is working on eliminating car accidents, but as a result, we might also eliminate traffic for good. Accidents are one of the leading causes of traffic congestion; just one fender bender could hold up miles of highway hours after the incident takes place. But traffic is also the result of "invisible accidents." These are near-misses, or can even be the result of a driver braking too hard through a curve. No human being can drive perfectly 100 percent of the time, and these small, seemingly insignificant driving mistakes add up over time. As our cars become more in-tune with the world around them through our ADAS features, we could take human error out of the equation entirely. Additionally, vehicle-to-vehicle communications will play a greater role in traffic navigation. As vehicles communicate with one another, they will have the power to adjust speed or change routes in order to clear a traffic jam entirely.

ADAS features are better than human judgment because they constantly scan the road and correct the driver's inputs. A distracted driver might fail to notice brake lights up ahead, but an ADAS-equipped vehicle will notice and slow down in advance. Last year, Tesla's autopilot software prevented this exact type of accident, braking for its driver within milliseconds. Its response time was far quicker than the average human eye could catch. In the very near future, we could see our cars surpass the response times of even the most talented human drivers.

Which to insure, the driver or the manufacturer?

YOUR INSURANCE PREMIUMS COULD DECREASE

Our industry is about to have a major impact on insurance premiums for car owners -- in fact, we might make individual insurance plans obsolete. Insurance companies decide on a premium price based on the car's safety features and the driver's accident history and overall demographic risk (such as how long they've had a driver's license). With new advances in ADAS features, the driver's capability will become less of a safety factor. ADAS systems are capable of overriding driver commands in the interest of safety, lowering the risk of accidents. That means that our customers will pay less every year to own their cars.

Additionally, insurance companies can see detailed diagnostic reports in the event of an accident to determine who or what, exactly, was at fault. Because insurance companies are taking less of a risk, and there's greater safety accountability, premiums will likely go down for individuals and car manufacturers over the next few decades. Experts still aren't sure whether insurance companies will insure the driver or the manufacturer in the case of truly autonomous vehicles, but either way, increased safety will result in lower insurance costs. After all, if driving becomes a virtually accident-free activity, then who needs an extensive insurance plan?

In the automotive industry, we're changing the entire social structure around car culture and transportation. As more car manufacturers use compilers like TASKING to build their ADAS code, we'll see changes in the way that insurance companies approach coverage, as well as how efficiently our cars move from point A to point B, all of which will make driving a more pleasant, and safe, experience.

INLINE FUNCTIONS IN C LANGUAGE: WHY AND WHEN THEY ARE USED IN ADAS SOFTWARE

Years ago I helped to re-roof a house, and the first task was to remove the old shingles. Pulling up shingles one at a time, I was very proud of my efforts until a professional roofer came over to me shaking his head. Using a scoop shovel like a snow plow, he removed a couple a hundred shingles in less time than it took me to pull up five by hand. I learned up on the roof that day to always look for better ways to get the job done.

When routing a PCB design, we also want a more efficient way to get the job done. You as a designer are probably aware of how much time is takes to manually get the cleanest route possible. You're probably also aware of Auto routers, which will do the job faster, but often with undesirable routing results. Now, like how the shovel was a better way to remove shingles, we have a better way to route using an auto-interactive router.

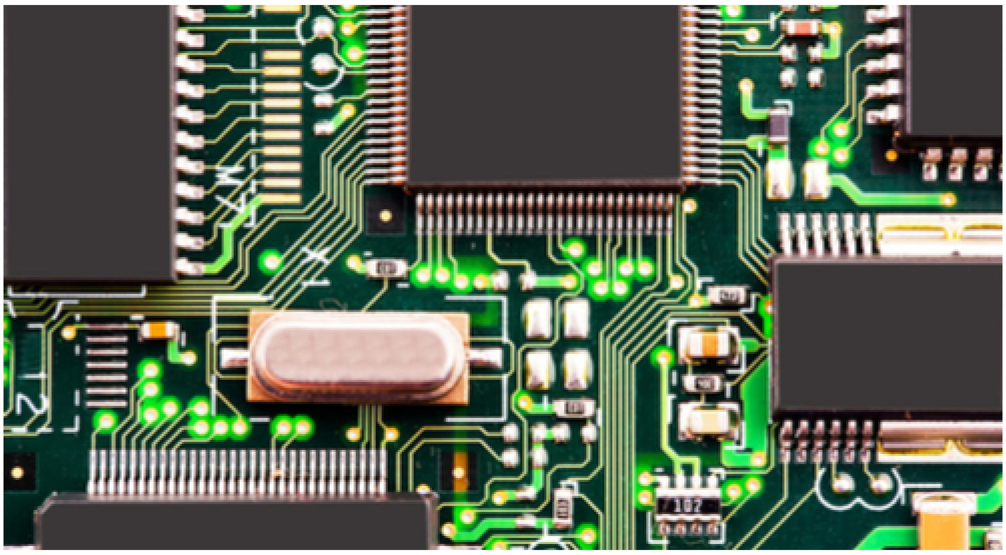
For those of you who aren't familiar with the term auto-interactive router, this is a relatively new routing technology. It is designed to give you trace routing that has the look and precision of manual routing but at approximately one connection per second. Auto-interactive routing is not a batch auto router and is not designed to route your entire board. Instead, it is interactive PCB auto routing where you have the control and the router does the work.

While auto-interactive routing can generally speed up your process, there are areas of circuitry that benefit from it. Particularly, single or multiple nets that need to be connected in a specific order without changing layers. More specifically this can be, point-to-point routing, BGA breakout routing, and bus routing.

POINT-TO-POINT TRACE ROUTING

As you know, PCBs have many point-to-point connections. These can be short nets that connect component pins in tight circuitry groups, via escape patterns from SMT pads or nets that run the length of the board. These are not difficult to manually route, but they take time. An auto router will route them very fast, but you may be left with undesirable routing on the rest of the board. You could restrict the auto routing to just the short connections, but that requires additional setup time before you can use the auto router.

An auto-interactive router is incredibly fast when routing these same nets. You will simply select the nets that you want to route and then engage the router. Using the design rules that you have already established in the database, the auto-interactive router will use the shortest possible path to route the traces. It will also do this without the cumbersome setups that an auto router requires.



An auto-interactive router works well to connect short point-to-point connections.

BGA TRACE ROUTING

Before manually routing traces out of a BGA, you'll need to take the time to figure out the best connection order of the nets. This ordering is important so that the designer can route the traces in the most efficient patterns possible to connect the BGAs. Efficient routing patterns not only leave open space on the board for additional routing but will reduce or eliminate the need for vias. Vias take up space that you may need for other routing, and they can introduce undesirable signal characteristics. Using an auto router for BGA routing may actually increase your via count as it typically does not choose the best connection order of the nets.

This is where using an auto-interactive router can be handy. It can quickly order the BGA connections for you. It will evaluate your selected nets in order to evenly distribute the traces on the board layers that you have enabled for routing. The auto-interactive router will then route the traces to connect the BGAs together in clean and precise routing patterns without using vias. These clean routing patterns will give you the additional board space that you will need for trace tuning or other routing edits later on.

Bus routing is another strength of an auto-interactive router.

BUS TRACE ROUTING

Often times buses are manually routed to create a clean routing pattern. If the bus is not routed in a clean pattern, it could cause problems with the signal characteristics of the bus. The downside is that manually routing a clean bus pattern takes a lot of time. Buses can have a lot of nets in them, and some boards may have several different buses as well. For example, double data rate (DDR) circuitry is an example of multiple bus routing with high net counts.

An auto-interactive router will route the nets of a bus for you, but it will do it under your control. First, you will select the nets of the bus that you want to route. Next you will draw a path on the board that you want the bus routing to follow. This path is a template that specifies the direction and width of the bus to the router. The auto-interactive router will then follow the path to route the traces all while obeying the design rules at the same time. The router does this work very quickly, and the results are a very clean bus routing pattern.

Just as the shovel was a much better way to remove shingles, an auto-interactive router is a much better way to route your PCB design. Knowing the best ways to use this router will be the key to your success, and we've talked about three of those ways. Point-to-point, BGA, and buses are all areas of routing that an auto-interactive router can help you to do a better job.

Routing is part of the PCB design process that Altium can help you to do a better job with. Altium Designer 18 is PCB design software with powerful resources, including auto-interactive routing with ActiveRoute. Would you like to find out more about how ActiveRoute will be able to help you to route your next PCB design? Talk to an expert at Altium.

INLINE FUNCTIONS IN C LANGUAGE: WHY AND WHEN THEY ARE USED IN ADAS SOFTWARE

I love watching "behind the scenes" clips of movies. It's fascinating how the actors, cameramen, and all the other crew work together to make a film. It's great to see the hilarious outtakes too. Seeing the little mistakes and unglamorous side of movies helps me to better appreciate them on the big screen. While compilers are not as entertaining as films, they also have a lot going on behind the scenes. It's easy to use compiler optimizations while having little understanding of what the compiler is actually doing to your code. However, understanding how these tools work can help you use them more effectively. Tools like function inlining can help speed up your program but may use extra space. That's why you need to carefully choose which sections of code to inline. If you don't, you might end up using extra space without increasing your performance.

WHAT ARE INLINE FUNCTIONS?

If a movie ends up being too long after filming, scenes might get trimmed or replaced in post production. In the same way, function inlining allows you to cut down the time it takes to call and return functions after you've already written the code. Inlining functions can take up more space than calling functions, but sometimes the size difference is minimal.

So what exactly is function inlining? Normally you call functions. The program then goes to that function, performs its operations, and returns a value or values. When functions are inlined, the compiler substitutes the function itself for the call of the function. This can speed up your software because you don't waste time calling a function and then returning its values. You simply run the function in line with your code. The way compilers do this is a bit complex, as your software should not be able to tell whether the function was inlined or not. Another important thing to know is that tagging something with "inline" does not necessarily mean the compiler will actually inline it. You just have to do your best to "suggest" to the compiler what needs to be inlined.

Function inlining can help you optimize your software.

Just like loop unrolling, function inlining brings up the classic time vs. space conundrum. You need to find the delicate balance between the two when programming for advanced driver assistance systems (ADAS). You need things like automatic braking to work quickly, but you also need to keep your program small enough to fit on your hardware. When your compiler inlines a function, it actually replaces your call lines with the lines of the function. If the function itself is longer than the call, it will take up more space. Sometimes functions are approximately the same size as their call, meaning they don't take up much space when inlined. Other times functions are much larger and will significantly increase the size of your code. If the function is called multiple times, your compiler might inline multiple copies of it, wasting space. That's why you always need to carefully choose which sections of code you want to optimize to make sure you don't trade too much space for speed.

WHICH FUNCTIONS TO INLINE

There are two main types of functions you should try to inline if possible: short functions and static functions. In the opposite direction, you should probably not inline long functions or ones that are repeated often.

Short functions are the low hanging fruit of inlining. If a function is about as many lines long as its call, you should definitely inline it. The compiler will replace the call with a function that's relatively the same size but gives you a speed boost. Static functions are also great to inline. They are especially great to inline when they have only one call. There are several other more complex advantages to inlining functions, but they are beyond the scope of this article. Inlining short and static functions will help your program execute operations in less time, without adding too much size to your code.

The primary disadvantage of inlining functions is that it takes up more space. Especially if the function you're inlining is quite long. Remember that you're putting the function directly into your code. If the function is extremely lengthy, you could be adding more lines than is helpful. Functions that are called very often can also be a problem. You don't want to inline those functions because you're essentially copying the same function over and over into your code. Space is limited in cars and on their microprocessors, so you don't want to waste space making copies of copies.

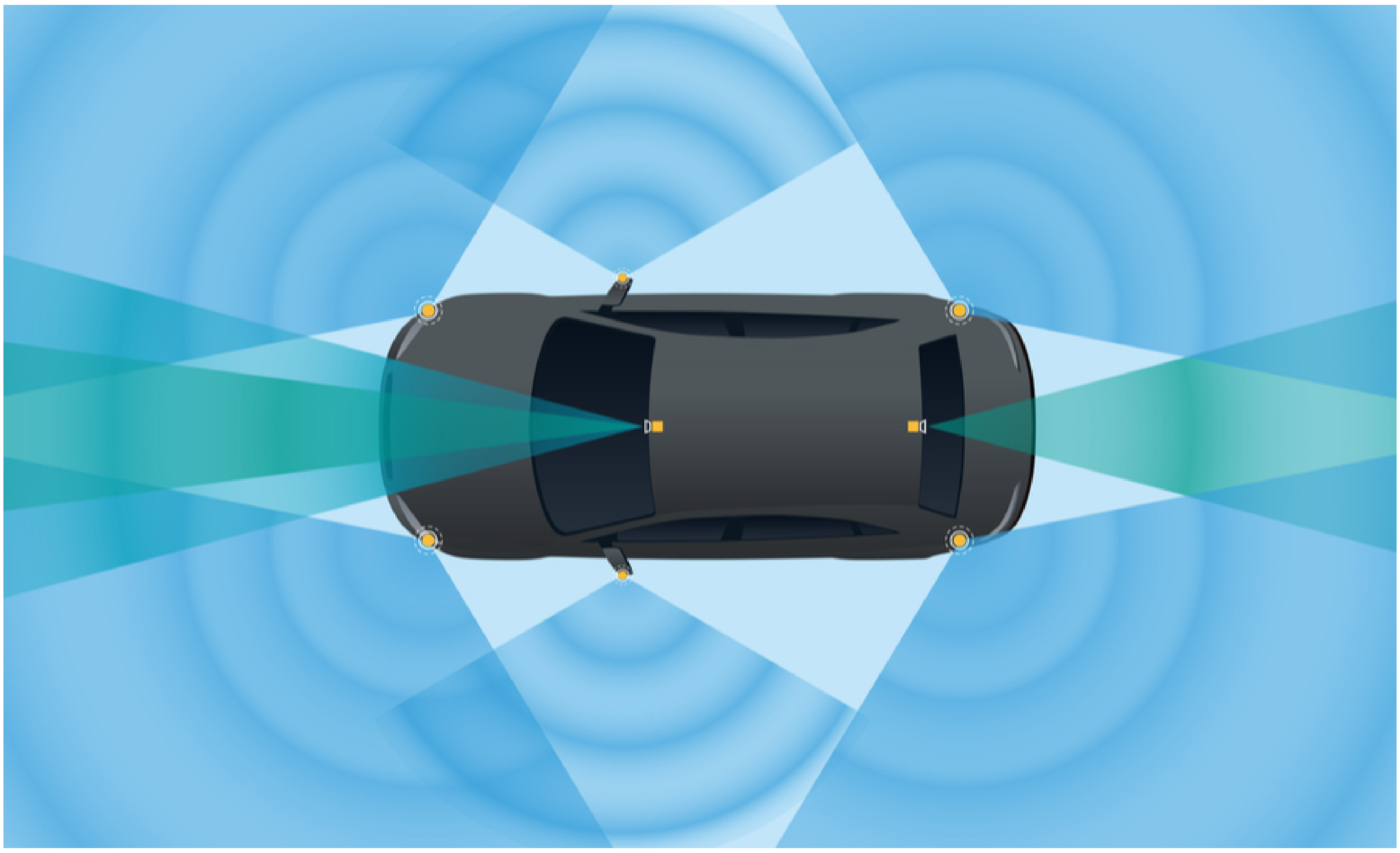
ADAS systems need optimized programs to operate efficiently.

Function inlining comes into the complex problem of code optimization. You can execute some sections of your program faster, but those sections will also be larger. That's why it's important that you accurately hint to your compiler which parts should be inlined. You should try to target short functions and static functions for inlining, while avoiding long functions or functions are used lots of times.

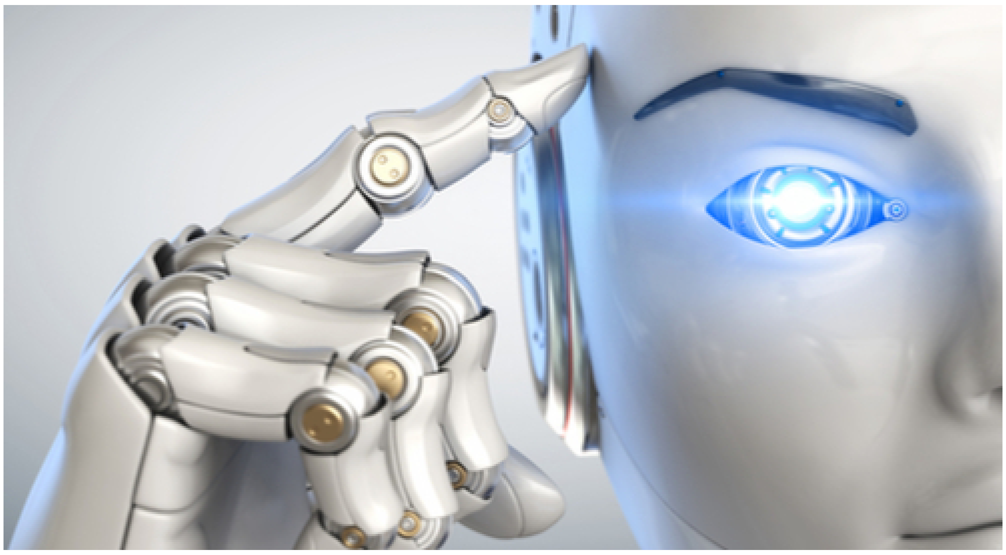
The thing about function inlining is that it should be done by a compiler. I would suggest using TASKING's compiler. TASKING also makes a wide variety of other tools that can help speed up your work.

Have more questions about function inlining? Call an expert at TASKING.

NEURAL NETWORK SUPERVISION AND ADAS VEHICLE SAFETY



Have you ever wished you were a bit smarter? I know I have, particularly during my linear signals and systems classes in college. Now we're using the things I learned in that class, like fast Fourier transforms, to make cars more astute. Neural networks can use a variety of incoming data to help automobiles learn and "see" the world around them. The only problem with giving machines intelligence is that it opens them up to failure, which I managed to avoid all those years ago. If vehicles can think and deduce, it means they could possibly make an incorrect decision. This possibility has to be addressed before cars with advanced driver assistance systems (ADAS) powered by neural networks can hit the roads. Luckily we can use conventional logic and sensors to supervise neural networks as they process data and determine actions.



Artificial Intelligence is the future for vehicles.

THE ADVANTAGES AND DISADVANTAGES OF NEURAL NETWORKS

The inherent advantage of neural networks is that they can remember how to react to old situations yet are flexible enough to deal with new ones as they arise. Unfortunately, this strength is also a weakness, as that flexibility means that we cannot be certain how they will react to novel inputs. In terms of safety, this makes neural networks hard to certify and not suitable for critical safety operations.

Machine learning is obviously a big deal for ADAS enabled vehicles. It's such a game changer that some chip companies, like NVIDIA, are pivoting to develop artificial intelligence (AI) chips specifically for the automotive industry. The reason that companies are rushing to implement AI in cars is that it could help them drive more accurately and safely. Google's autonomous cars have been navigating the roads for years, but the only reason they can is because Google has preprocessed maps of the paths they travel. A car with artificial intelligence wouldn't need to drive on a road that had already been mapped out with precision. I know that whether or not there is a painted line on a road, I should be driving on the right side. A car that can "think" should be able to deduce the same thing. I also know I can cross a solid centerline in order to avoid a wreck. A vehicle that relies solely on rules and traditional sensors may not be able to make that decision. A system that can learn should be able to follow or break the rules in order to maintain its safety.

There are multiple impediments to implementation of machine learning in cars, but one that's often forgotten is safety. There is no way to guarantee that a neural network will always make the right decision by itself. There have been times where I thought I was good to pass, pulled out into the other lane and had to dash back quickly to avoid an oncoming car. Some decisions need to be made with 100% certainty. People can learn things the wrong way, are prone to thinking that correlation implies causation, and can assume things will always be as they have been. For example, in my state when turning right at an intersection you must turn into the closest lane. Thus people coming in the opposite direction can simultaneously turn left into the lane next to you. In other states, it's common practice to turn right into any lane you please. I once tried to turn left onto a street while another person was turning right onto it at the same time and we nearly had a wreck. A car with AI might make an assumption either way and end up crashing into someone.

Make sure your car thinks like a good driver and not like a child.

NEURAL NETWORK SUPERVISION

It seems obvious that these situations are avoidable, and they are. You can't let a neural network make decisions alone, but you can let it suggest actions that are verified by a supervisor. Conventional methods that can be safety certified can be used to validate a neural network's suggestions.

Let's take that previous example again. Your car is about to turn left into a leftmost lane that should be clear, as anyone turning right will turn into the rightmost lane. You start to turn and another car turns right into the leftmost lane. Your vehicle's sensors should pick up the impending collision and stop your car, even though you "should" have been able to turn into that lane. Traditional sensors can set hard limits on a neural network's decisions by checking against some basic rules. Is the space we are trying to enter occupied, are we accelerating while the car in front of us is braking, etc. In order to accurately check your neural network's deductions, you will need to be certain that your conventional system works. For that, I recommend using multiple sensor fusion to combine data streams for decision making. Thus you can pair your neural network, which will be rated at a relatively low ASIL level, with a system rated at the highest level for checking.

The advantages that neural networks have to offer are too great for us to ignore the complexity of their implementation. Soon manufacturers across the board will be using AI in their ADAS enabled vehicles. When that time comes make sure that your system has a conventional check in place to ensure that your neural networks are suggesting reasonable actions.

Building a car with ADAS is no simple task, whether it uses AI or not. If you want to do it right you need software that's made specifically for developing programs for vehicles. TASKING has that software. They have a huge range of products, from standalone debuggers to static analysis tools, that will ease the pain of development.

Have more questions about machine learning? Call an expert at TASKING.

ADDITIONAL RESOURCES

Thank you for reading our guide on Auto-Interactive Routing. To read more Altium resources, visit the Altium resource center here or join the discussion at the bottom of each original blog post:

- Growing ADAS Market Will Require Efficient Software and MPU Expertise
- Augmented Reality in Cars: Pros and Cons of ADAS Heads up Displays
- Which ADAS Features Are Most Likely to Lower Premiums and Increase Public Safety?
- Inline Functions in C Language: Why and When They Are Used In ADAS Software
- Neural Network Supervision and ADAS Vehicle Safety