



# Design Considerations for Autonomous Mobile Robots



While autonomous robots have long seemed an idea out of science fiction, today they are a real technology being deployed at scale. And, within the robotics field, one of the fastest growing technologies is Automated Mobile Robots (AMRs).

AMRs are self-navigating robotic systems built to operate without human intervention in warehouse logistics environments. While Automated Guided Vehicles (AGVs) follow predefined routes using beacons or tracks, AMRs rely entirely on onboard sensors, real-time processing, and AI to interpret their environment and make autonomous decisions. This autonomy allows them to adapt on the fly, navigate around unexpected obstacles, and dynamically update task priorities.

The computational architecture underlying AMR behavior has evolved significantly since the field's inception. Early systems used a centralized processing model where a high-performance processor gathered and interpreted all sensor data, then made decisions for motion, mapping, and control. While effective, centralized systems introduced latency and power inefficiencies, especially as the number and complexity of sensors increased. These limitations become pronounced in applications that demand real-time responsiveness, energy efficiency, or modular scaling across fleets.

To overcome these challenges, modern AMRs increasingly adopt a distributed architecture. Edge processors embedded within vision systems, motion controllers, and other subsystems now handle local computation. This approach reduces the workload on the central processor to unlock lower-power designs and faster real-time decision-making.

This white paper presents an overview of design considerations across AMR subsystems and explores how NXP's scalable processor portfolio and software ecosystem can support each function.

## **A look inside AMRs: subsystems and design considerations**

Tasked with autonomously navigating through dynamic environments, AMRs rely on a complex set of interconnected hardware systems. On the highest level, the hardware within an AMR can be broken down into the following subsystems

- Sensors and perception
- Compute and navigation
- Battery management
- Motor control and actuation

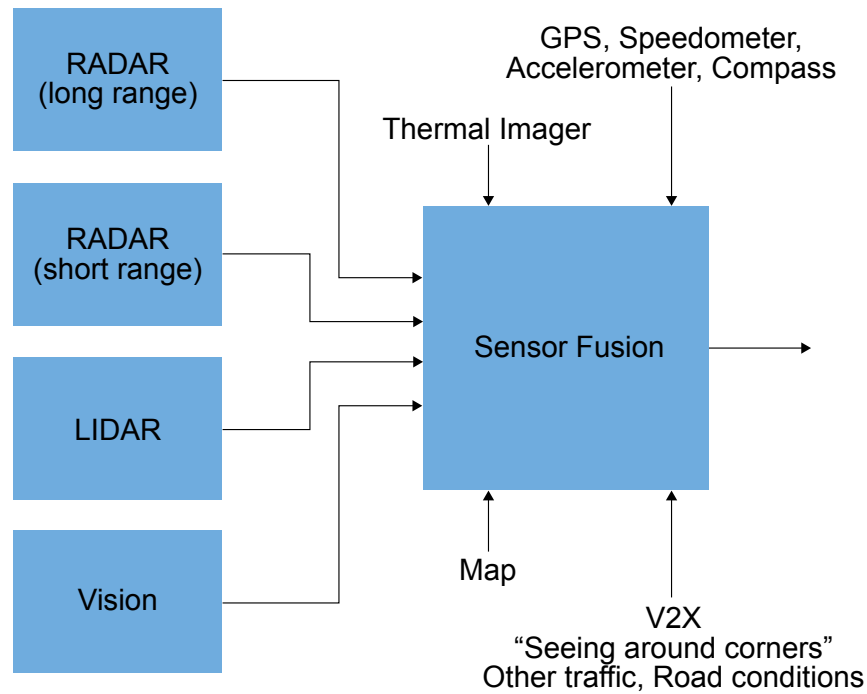
### **Sensors and perception**

Sensors define how an AMR perceives its environment. To operate autonomously, a robot must detect and classify objects, estimate distances, understand motion, and localize itself within a map. This level of awareness depends on a combination of sensing modalities, each of which offers unique benefits. The major challenge for the designers lies in selecting the appropriate sensors and fusing their outputs into a coherent model that supports accurate, real-time decision-making.

#### **Vision sensors**

Vision sensors are the foundation for object recognition and environmental classification. RGB cameras capture two-dimensional color images used in machine vision algorithms for tasks like pallet detection and obstacle recognition. However, 2D data lacks depth, which limits its effectiveness for spatial planning. To resolve this, AMRs integrate stereo or 3D cameras, which estimate depth by comparing the disparity between two images. This supports fine-grained spatial reasoning, such as recognizing object contours or measuring free space.

An alternative approach involves Time-of-Flight (ToF) cameras, which calculate distance by measuring how long emitted light takes to return after striking a surface. ToF systems are compact and offer low-latency depth data.



**Figure 1:** Multiple different sensors in an ADAS system

## LiDAR

LiDAR (Light Detection and Ranging) is fundamental for mapping and obstacle detection. It works by emitting laser pulses and measuring the time it takes for each pulse to reflect off a surface and return to the sensor. The time-of-flight data is then used to calculate precise distances and create a detailed 3D map of the surrounding environment.

2D LiDAR scans a horizontal plane to detect obstacles, often triggering safety functions like slowdowns or emergency stops when a hazard is detected. 3D LiDAR extends this capability by generating volumetric point clouds that enable full environmental reconstruction and object localization. Due to cost constraints, some developers tilt 2D LiDAR units to approximate 3D coverage. The tradeoff ultimately lies in balancing resolution with budget.

## Inertial measurement units

Inertial Measurement Units (IMUs) provide information about motion, orientation, and acceleration. They do this by measuring acceleration, angular velocity and sometimes magnetic fields using a combination of accelerometers, gyroscopes, and magnetometers. When fused with vision or LiDAR data, IMUs help AMRs maintain stable localization even in areas with limited visual features. Many embedded cameras also include IMUs to enhance pose estimation during movement.

## Sensor fusion

While each sensor is useful on its own, AMRs truly become powerful when data from multiple sensors is fused together. In this effort, raw data from cameras, LiDAR and IMUs must be filtered, synchronized, and processed to form a unified perception layer. Edge AI is essential in this context. By executing preprocessing tasks like object detection and depth estimation within the sensor subsystem itself, edge AI reduces data volume and latency in the communication pipeline. Such a distributed approach improves system responsiveness and enables the use of lower-power central processors, which decreases costs and improves efficiency.

NXP tackles the need for edge intelligence through its lineup of application processors. The [i.MX 8M Plus](#) integrates a neural processing unit (NPU) for real-time AI inference directly on image data. It focuses on machine learning and vision for industrial automation and is built to meet the needs of Smart Home, Building, City and Industry 4.0 applications.

The [i.MX 95 family](#) delivers safe, secure, power-efficient edge computing for automotive edge, commercial IoT and industrial platforms. It combines powerful AI-accelerated vision processing with functional safety, advanced security and high-performance connectivity

For cost-sensitive applications, the [i.MX 93](#) delivers efficient machine learning (ML) acceleration and advanced security to support energy-efficient edge computing.

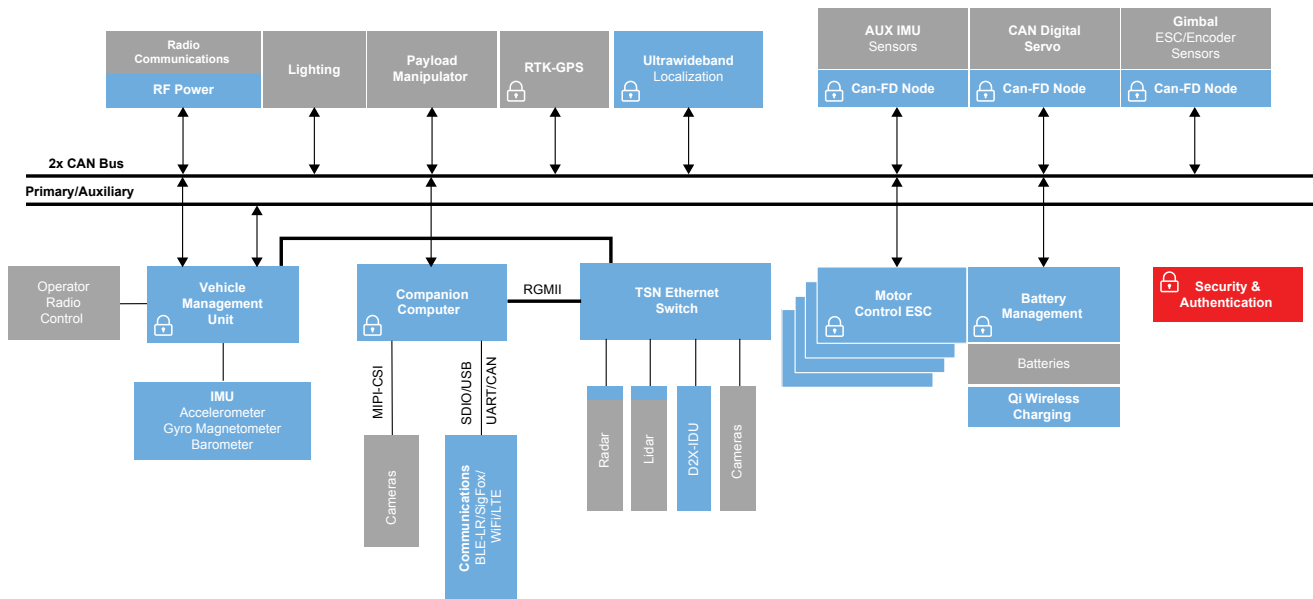


Figure 2: Modular mobile robotics architecture block diagram

## Compute and navigation

Autonomous navigation in AMRs depends on a combination of mapping, motion planning, and decision-making workloads, all of which must run in real time. These functions allow the robot to understand its surroundings and dynamically respond to changes in the environment.

Central to this process is Simultaneous Localization and Mapping (SLAM). SLAM algorithms construct a map of the robot's environment while simultaneously estimating its own position within that map. Traditional SLAM implementations rely heavily on LiDAR data to determine structure and range, but Visual SLAM (vSLAM) uses image features from cameras to improve localization accuracy. vSLAM is often used in environments where LiDAR returns are sparse or ambiguous, such as areas with glass or featureless surfaces.

Once the robot has used SLAM to establish its position and built a representation of its environment, it must determine how to move. Path planning and obstacle avoidance algorithms analyze this map in real time by calculating optimal routes while accounting for both static and dynamic obstacles. These algorithms must evaluate conditions such as wall placement and the unpredictable movements of people or other mobile platforms. Generally, navigation performance is measured by both accuracy and the robot's ability to respond quickly and safely under shifting conditions.

In modern AMRs, AI-based decision-making enhances the process by learning from previous interactions. Deep learning models trained on motion and environmental data can recognize patterns and

inform adaptive behaviors. Within this, neural networks can refine routing preferences and adjust the robot's course based on contextual information rather than hard-coded rules.

Traditionally, these workloads ran on a centralized compute model, where a single high-performance processor handled all vision, mapping and navigation tasks. However, this model introduces significant power consumption and system complexity, especially in platforms with rich sensor arrays. In contrast, modern AMRs are moving toward distributed compute architectures, where individual sensor nodes perform local processing and transmit processed insights to a more modest central processor. This architecture allows engineers to develop modular, scalable robot designs that meet application needs without over-provisioning compute resources.

NXP provides multiple processor options to support this evolving architecture. The i.MX 95 with six cores is well-suited for high-performance navigation and sensor fusion thanks to a combination of advanced AI capabilities and multi-core processing, advanced security, and safety. The i.MX 8M Plus supports real-time edge inferencing for vision-based navigation and lightweight SLAM. The automotive family of processors [S32G](#) and [S32N](#) provide multi-core compute combined with ASIL-D level of safety.

For deterministic control loops and low-latency motion execution, the [i.MX RT crossover MCUs](#) offer real-time responsiveness with integrated control and connectivity features. For MCUs with automotive safety ASIL-D capability, the [S32K](#) family provides a scalable family of MCUs.

## Battery management

Like in most mobile applications, energy management is a central design concern in AMRs. Battery capacity directly limits runtime, which in turn affects task scheduling, throughput, and labor efficiency. Therefore, the longer an AMR can operate between charges, the more useful it becomes. Battery performance also influences the total cost of ownership, including the number of charging stations required and the frequency of battery replacements.

To maximize operational uptime, AMRs must incorporate sophisticated battery management systems (BMS) that monitor, protect, and optimize power usage in real time. The goal is to maintain peak performance without compromising safety or battery longevity. Ultimately, a well-designed BMS ensures that high-power subsystems receive adequate energy while minimizing unnecessary draw during low-demand periods.

## Monitoring and protection

Modern battery systems rely on embedded controllers to continuously track voltage, current, temperature, and charge-discharge cycles. These metrics help detect early signs of degradation or thermal runaway and prevent damaging conditions like overcharging, overheating or deep discharge. Integrated protection mechanisms can isolate the battery or reduce system load if thresholds are exceeded.

## Power optimization

Beyond protection, AMRs benefit from power optimization techniques that adjust performance based on operational context. Dynamic power scaling allows processors to throttle frequency and voltage based on workload intensity. For example, during idle periods or between missions, non-essential components can enter low-power states. Likewise, wake-up modes allow the system to resume full operation without the need for a full reboot.

[NXP's MCX microcontroller portfolio](#) is specifically designed for ultra-low-power operation. With a compact footprint and integrated analog peripherals, they provide the real-time responsiveness needed to track battery parameters and apply power-saving strategies at the system level.

[NXP also offers reference designs](#) for battery management that integrate protection circuitry, communication interfaces, and embedded algorithms for charge estimation and health diagnostics. By leveraging these building blocks, developers can shorten design cycles while developing power systems that meet regulatory and performance standards.

## Motor control and actuation

Mobility in AMRs depends on accurate and reliable motor control. Whether the platform is maneuvering a tight warehouse aisle or aligning beneath a conveyor for loading, accurate actuation is necessary for the AMR to operate autonomously with minimal mechanical error.

AMRs typically use one of three motor types, each selected based on application demands.

- **Brushless DC (BLDC) motors:** These motors operate using electronic commutation, where switching current through stator windings produces smooth and continuous rotation. They offer the benefits of high efficiency, long service life and low maintenance.
- **Stepper motors:** These motors divide a full rotation into a set number of precise steps by energizing coils in a specific sequence, allowing exact position control without feedback. Such precision makes stepper motors a great choice for tasks that require incremental motion, such as robotic arms or lift mechanisms.
- **Servo motors:** These motors use a feedback loop to continuously adjust motor input based on real-time position or speed data. Servos are best for AMRs that require adaptive movement or coordinated actuation.

However, Effective motor control requires more than basic actuation; real-time responsiveness is crucial. The motor control system must adjust speed, direction, and braking within milliseconds. Such behavior requires deterministic response times and minimal control latency, especially when multiple motors are operating in coordination.

To guarantee smooth and accurate motion, AMRs employ closed-loop control systems that rely on continuous feedback from rotary encoders, wheel tachometers and IMUs to monitor velocity and position. With sensor feedback, control algorithms compare actual performance to commanded trajectories and adjust in real time to correct for any deviations.

Meanwhile, advanced AMRs and humanoid robots often demand multi-axis synchronization, or the coordinated control of multiple motors to move in precise relation to one another in both time and space. Given the timing requirements associated with it, such a feat necessitates industrial-grade communication protocols. For this reason, AMRs might rely on protocols like EtherCAT, which is a real-time Ethernet-based fieldbus with extremely low latency.

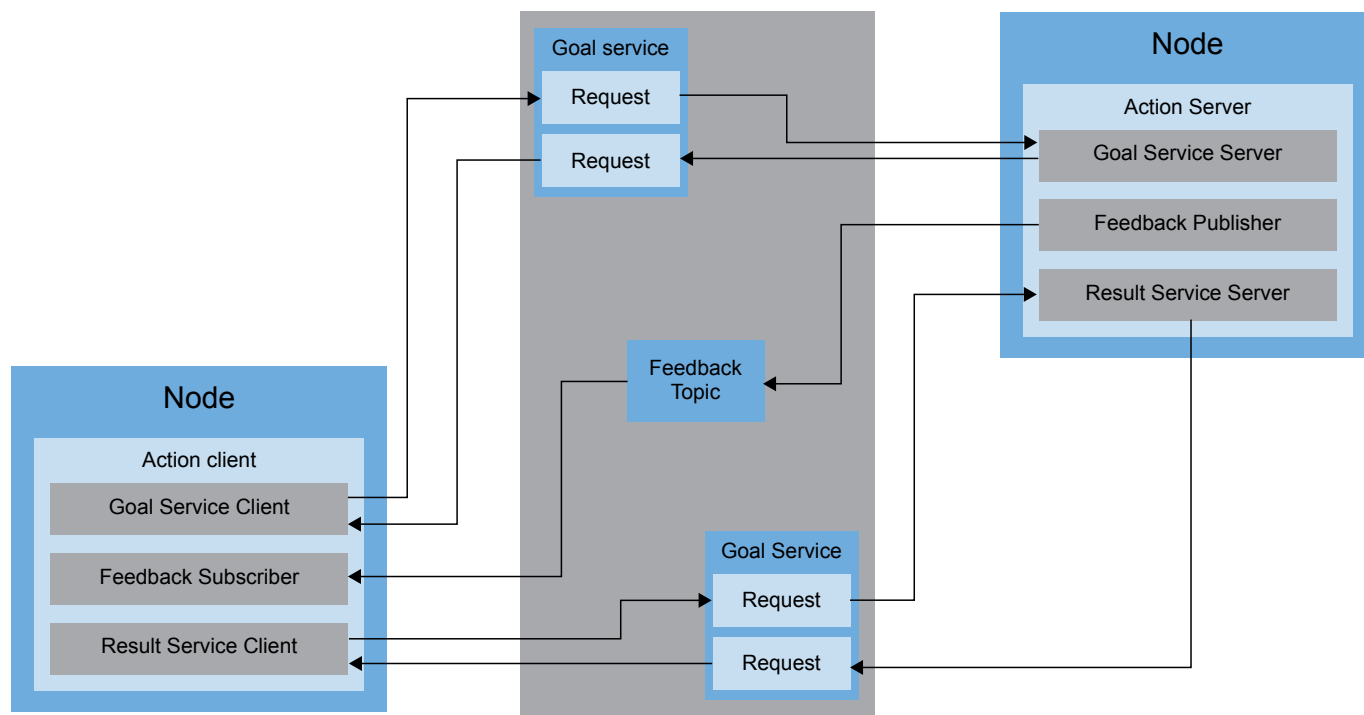
Fortunately for the designer, NXP provides scalable motor control solutions for a wide range of AMR platforms. For example, the [i.MX RT1180 crossover MCU](#) is designed for high-end motor coordination and supports EtherCAT natively. For most AMRs, NXP's MCX series offers an efficient and cost-effective solution. These microcontrollers can be deployed per motor or per axle for a modular architecture that scales easily with system complexity.

### Software enablement and ecosystem

While hardware is extremely important for AMRs, the underlying software stack is just as vital. From motion planning to perception and control, the software determines how effectively the robot interacts with its environment. To manage this complexity, developers need modular, open-source platforms that facilitate development and guarantee compatibility across hardware subsystems.

As such, ROS 2 (Robot Operating System 2) has become the de facto middleware for AMR development. Specifically, ROS 2 provides a flexible framework for message passing, real-time control, and hardware abstraction. With it, developers can tap into a wide array of prebuilt libraries for sensor drivers, navigation algorithms, and simulation tools, all of which dramatically accelerate prototyping and reduce the need to build functionality from scratch. ROS 2 also supports distributed system architectures, allowing sensor and actuator nodes to operate independently while staying synchronized within the overall control system.

For vision and AI workloads, AMRs rely on robust computer vision and machine learning stacks. Here, tools like OpenCV are used to implement functions such as object detection and image segmentation. In parallel, AI frameworks, often leveraging TensorFlow Lite or ONNX, are used to run inference models for SLAM, obstacle recognition, and motion prediction. These models must be carefully optimized to run with low latency on embedded processors with limited thermal and power budgets. To that end, hardware acceleration via NPUs or GPUs can dramatically reduce inference time and improve system responsiveness.



**Figure 3:** Actions are one of the communication types in ROS 2 and are intended for long running tasks. They consist of three parts: a goal, feedback, and a result.

To simplify software integration and accelerate time to deployment, NXP offers a suite of reference designs. For example, the NavQPlus platform provides integrated compute solutions for vision and AI applications, including support for camera inputs, AI inferencing, and ROS 2 compatibility.

## Conclusion

Designing effective Automated Mobile Robots requires careful coordination across sensing, compute, power, and motion subsystems. Fortunately, NXP Semiconductors offer a broad portfolio of processors and microcontrollers that directly address these requirements. From AI-enabled vision processing with the i.MX 8M Plus and i.MX 95 to production-ready reference designs, software enablement tools, and ROS 2-compatible development platforms, NXP offers all the resources needed to build high-performance, application-specific robotics platforms.

To explore these solutions and accelerate your next AMR project, visit NXP's [mobile robotics page](#) for access to technical documentation, reference designs, and developer resources.

## How to reach us

**Home Page:** [nxp.com](http://nxp.com)

**Web Support:** [nxp.com/support](http://nxp.com/support)

### **USA/Europe or Locations Not Listed:**

NXP Semiconductors USA, Inc.

Technical Information Center, EL516

2100 East Elliot Road

Tempe, Arizona 85284

+1-800-521-6274 or +1-480-768-2130

[nxp.com/support](http://nxp.com/support)

### **Europe, Middle East, and Africa:**

NXP Semiconductors Germany GmbH

Technical Information Center

Schatzbogen 7

81829 Muenchen, Germany

+44 1296 380 456 (English)

+46 8 52200080 (English)

+49 89 92103 559 (German)

+33 1 69 35 48 48 (French)

[nxp.com/support](http://nxp.com/support)

### **Japan:**

NXP Japan Ltd.

Yebisu Garden Place Tower 24F,

4-20-3, Ebisu, Shibuya-ku,

Tokyo 150-6024, Japan

0120 950 032 (Domestic Toll Free)

[nxp.com/jp/support](http://nxp.com/jp/support)

### **Asia/Pacific:**

NXP Semiconductors Hong Kong Ltd.

Technical Information Center

2 Dai King Street

Tai Po Industrial Estate

Tai Po, N.T., Hong Kong

+800 2666 8080

[support.asia@nxp.com](mailto:support.asia@nxp.com)

[nxp.com](http://nxp.com)

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2025 NXP B.V.

Document Number: DESAMRWP REV 0