Motor control architectures for dexterous humanoid robot hands

Introduction

Dexterous hands are among the most complex and cost-intensive humanoid subsystems. They require the following:

- Coordinated actuation
- Position and contact feedback
- Calibration
- Communication
- Local processing

These functions must fit within strict limits on size, mass, wiring, power, and heat.

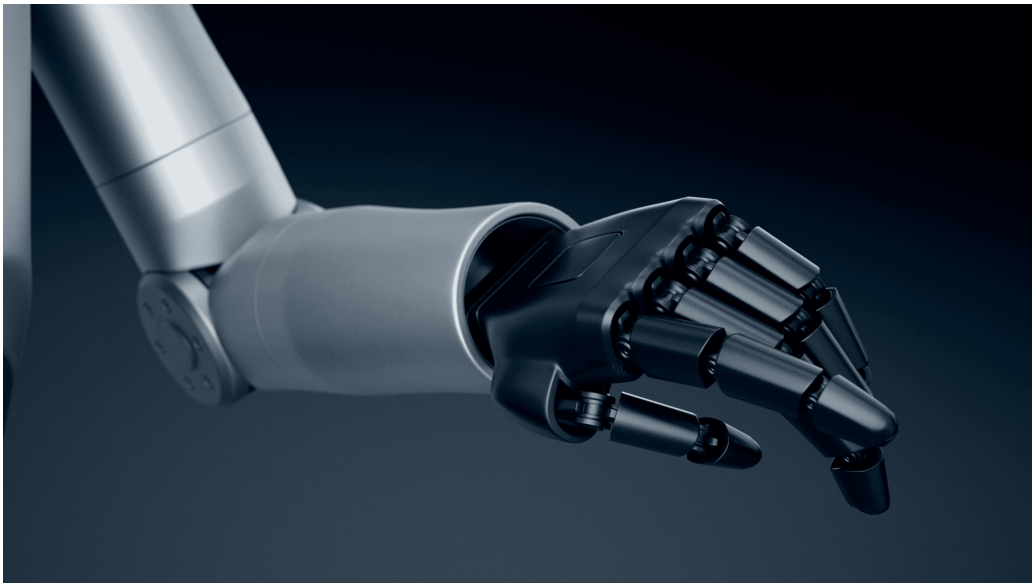
Adding degrees of freedom increases the mechanical capability of the hand and the potential range of motion. Dexterity also requires control software that coordinates the actuators and feedback channels with appropriate timing, calibration, and motion strategies. Palm-based, tendon-based, and in-finger, direct-driven architectures create different tradeoffs in moving finger mass, transmission behavior, electronics density, sensing, serviceability, and thermal design.

This application note describes three principal architecture types:

1. Palm-based architecture
2. Tendon-based architecture
3. In-finger, direct-driven architecture

It details the tradeoffs of each architecture, proposes suitable ST architecture directions in Table 3, and then highlights two palm-based ST reference architectures that scale from compact 6 degrees of freedom (DoF) control to a sensor-rich 16-DoF platform. For the 16-DoF platform, this application notes lists involved ST building blocs in Table 5.

Figure 1. Hand unit



1 General information

The devices STM32C5 series, STM32G4 series, STM32H5 series, STM32H7 series, and STM32N6 series have an Arm® Cortex® core.

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

Table 1. Reference documents

Document number	Title
[1]	*Motor control architectures for humanoid robot joints* (AN6569)

2 Glossary

Table 2. **Glossary**

Term	Meaning
DoF	Degrees of freedom
FOC	Field-oriented control
FSM	Finite state machine
IMU	Inertial measurement unit
ISPU	Intelligent sensor processing unit
MLC	Machine learning core
PWM	Pulse width modulation
SBB	System building block
SPI	Serial peripheral interface

3 The hand as a sensor-actuator subsystem

A humanoid hand is a distributed sensor-actuator subsystem, not simply a collection of small motors. Useful dexterity depends on the coordinated behavior of motor-control channels, mechanical transmissions, position and current feedback, contact sensing, communication, calibration, and higher-level hand control.

Actuator placement strongly influences electronic architecture. Palm-based designs concentrate motors and electronics in the palm, but available palm volume and the resulting wiring fan-out can limit practical motor-channel and finger DoF scaling. Tendon-based designs place motors in the palm, wrist or forearm, where more space may accommodate larger motors and improved thermal management. These architectures can support higher force or payload targets, but introduce elasticity, friction, hysteresis, and calibration challenges. In-finger, direct-driven designs improve modularity and shorten the mechanical path, but limited motor volume can constrain output torque and payload. These designs may need more motor power. They might also require faster motors with gear reduction. At the same time, they must meet strict size, wiring, and heat requirements.

Across these architectures, time-critical motor control and immediate electrical protection should remain close to the motors. One or more motor-control MCUs execute pulse-width modulation (PWM) generation, current acquisition, and position-feedback processing for the hand actuators. A separate bridge MCU can coordinate the motor-control nodes. It can aggregate sensor data and manage calibration. The MCU connects the hand to the arm controller through CAN FD, Ethernet, or an external EtherCAT slave controller. Optional perception processing may run on another processor or processing domain and should remain separated from the deterministic motor-control loops.

A complete hand sensor-actuator platform may include:

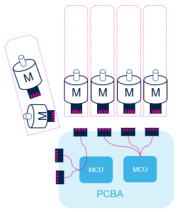
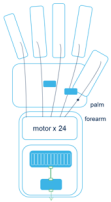
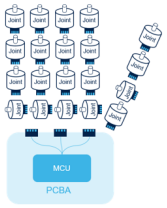
- Local motor-control channels with synchronized current and position acquisition
- One or more motor-control MCUs for deterministic actuator control
- A bridge MCU, or zonal control unit, for coordination, sensor aggregation, calibration, and arm communication
- One or two position sensors per controlled axis, with tactile, pressure, temperature or inertial sensing where needed
- Protected power distribution and persistent motor, transmission, and sensor calibration

Figure 2 illustrates the three main types of architecture types that create different tradeoffs. These tradeoffs affect the following:

- Finger mass movement
- Transmission behavior
- Electronics density
- Thermal design
- Wiring
- Sensing
- Serviceability

Finger-segment or fingertip inertial measurement units (IMUs) can enhance motor and joint encoders. They provide direct feedback on motion and orientation. This feedback is especially useful when linkages or tendons make motor-side position an incomplete representation of actual finger movement.

Figure 2. Actuator, transmission, and electronics placement in three humanoid hand architectures

	Palm-based (linkages)	Tendon-based	In-finger, direct-driven
			
Typical DoF	• 6 - 16	• 20 - 22	• 16 - 24 (fully actuated)
Force control potential	• High, with suitable joint or contact sensing	• Medium, due to tendon friction, elasticity, hysteresis	• Very high, with local feedback and short transmission path
Typical payloads	• Higher (5 - 20 kg)	• Lower (1 - 1.5 kg); focus on flexibility and leightweight	• Medium (2 - 5 kg)
Durability	• High (no maintenance for tendons)	• Medium (tendons are consumables)	• High
Cost	• Medium cost (standardized production)	• Medium-high (complex process)	• High (motors and control needs)
Heat dissipation challenge	• High on system level	• Medium on system level	• Low on system level but high for each individual actuator
Applications	• R&D, high-precision operation, industrial	• Service applications	• R&D, high-precision operation

DT80468V1

Table 3 summarizes the key actuator classes used in dexterous hand systems and highlights the corresponding architectural directions.

Table 3. Actuator classes and architecture directions for dexterous hand motor control

Architecture type	Palm-based	Tendon-based	In-finger, direct-driven
Actuator and motor direction	Coreless DC, BLDC, or compact frameless motors and motor-control electronics concentrated in the palm.	Coreless DC, BLDC, or frameless motors in the palm, wrist, or forearm. Tendons transmit force to the fingers	Miniature coreless DC, BLDC, or compact frameless motors located close to the finger joints.
Main design priority	• Multiaxis integration • Palm thermal management • Controlled wiring fan-out	• Low finger mass • Use of available wrist or forearm volume • Compensate for tendon friction, elasticity, and hysteresis	• Extreme miniaturization • Modular finger nodes • Local heat dissipation
Electrical envelope	Typically, 12 V to 24 V, while 48 V are emerging, commonly 5 A or less per motor		
Suitable ST architecture direction	STM32G4 series motor-control MCUs with compact STSPIN motor drivers. STM32H5 and STM32H7 series bridge MCUs for sensor aggregation and arm communication. RS-485, CAN FD, or EtherCAT as actuator communication network.	STM32G4 series motor-control MCUs with compact STSPIN motor drivers. STM32H5 and STM32H7 series bridge MCUs for hand coordination and arm communication. RS-485, CAN FD, or EtherCAT as actuator communication network.	Distributed STM32G4, STM32C5 series motor nodes (20-40 kHz PWM), or STM32H5 series (PMW towards 100 kHz). STM32H5, STM32H7 series bridge MCUs. STSPIN drivers, encoder/current feedback. RS-485, CAN FD, or compact local-bus as actuator communication network.

Architecture type	Palm-based	Tendon-based	In-finger, direct-driven
Emerging requirements	Higher channel and sensor density. Finger-motion sensing to complement palm-side motor feedback. Edge AI perception, scalable wiring, and thermal management.	Tendon-state estimation and compensation. Finger-motion sensing after the compliant transmission. Sensor-assisted grasp control (IMUs, pressure sensors) Organic material detection.	Higher power density. Dual position feedback. Optional IMUs where local encoders do not provide sufficient finger-motion context. Compact shared-bus communication.

Actuator placement determines much more than mechanical build. It influences feedback requirements, calibration effort, thermal management, wiring complexity, serviceability, and the distribution of processing resources across the hand.

Palm-based architecture

The palm-based architecture centralizes motors, drivers, and control electronics in one enclosure. This simplifies how fingers move. It also allows communication, sensing, and power resources to be shared across multiple channels. However, it increases the density of PCBA connectors and the thermal density in the palm.

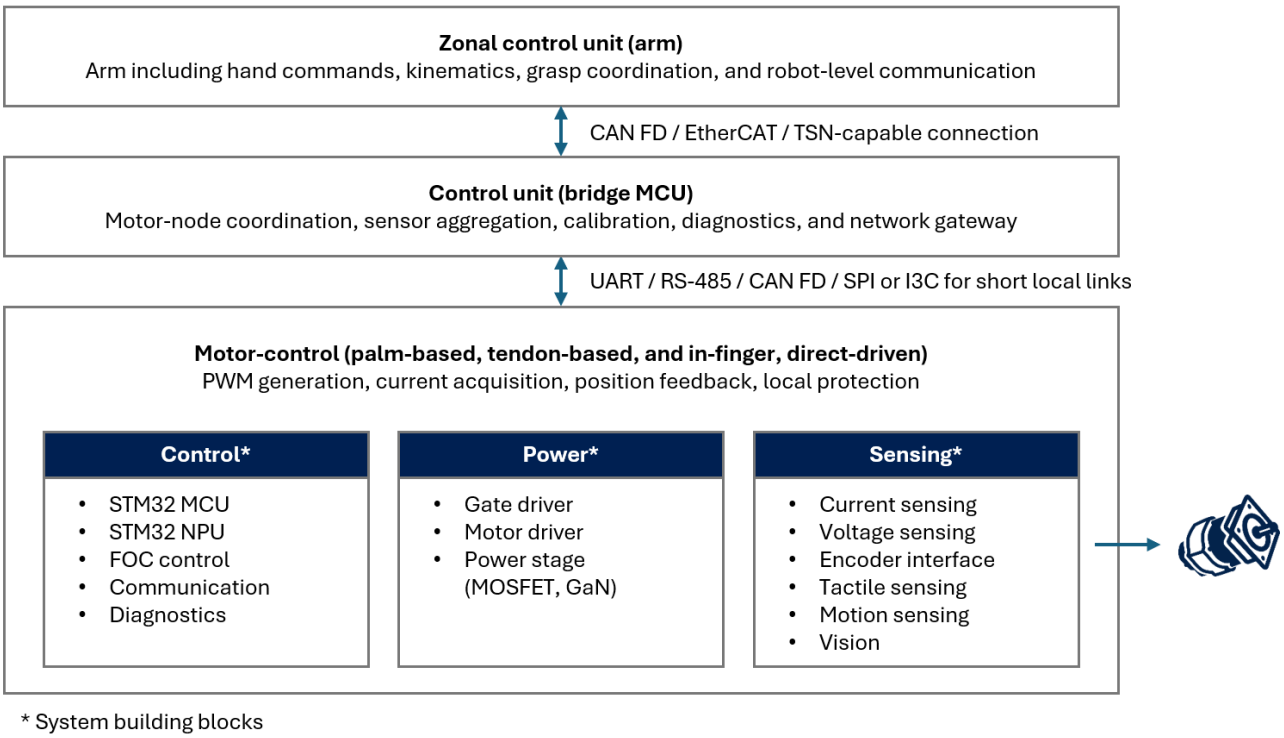
Tendon-based architecture

The tendon-based architectures place motors in the palm or forearm and transmit force through mechanical tendons. This reduces moving finger mass but introduces elasticity, friction, hysteresis, and routing-dependent behavior that often increase calibration and feedback requirements.

In-finger, direct-driven architecture

In-finger, direct-driven architectures place motors, and electronics close to the controlled joints. This improves modularity and shortens the mechanical path, but imposes the strongest constraints on motor size, wiring, local power distribution. Heat dissipation also plays an important role in this architecture. By distributing the motors to different locations within the hand, hot spots can be avoided. The smaller PCBs and the higher integration of the power stages increase the heat dissipation needs. This change helps to ensure reliable and continuous operation.

Figure 3. Partition between motor-control nodes, bridge MCU, and arm controller



* System building blocks

DT80469V1

This partitioning keeps deterministic motor loops in the motor-control nodes, while the bridge MCU coordinates the hand, aggregates sensor data, and provides the network connection to the arm controller. It prevents the bridge MCU from servicing every PWM and ADC event and limits the raw motor and sensor data transmitted across the robot network.

Table 4. System-level design priorities for humanoid hand motor drives

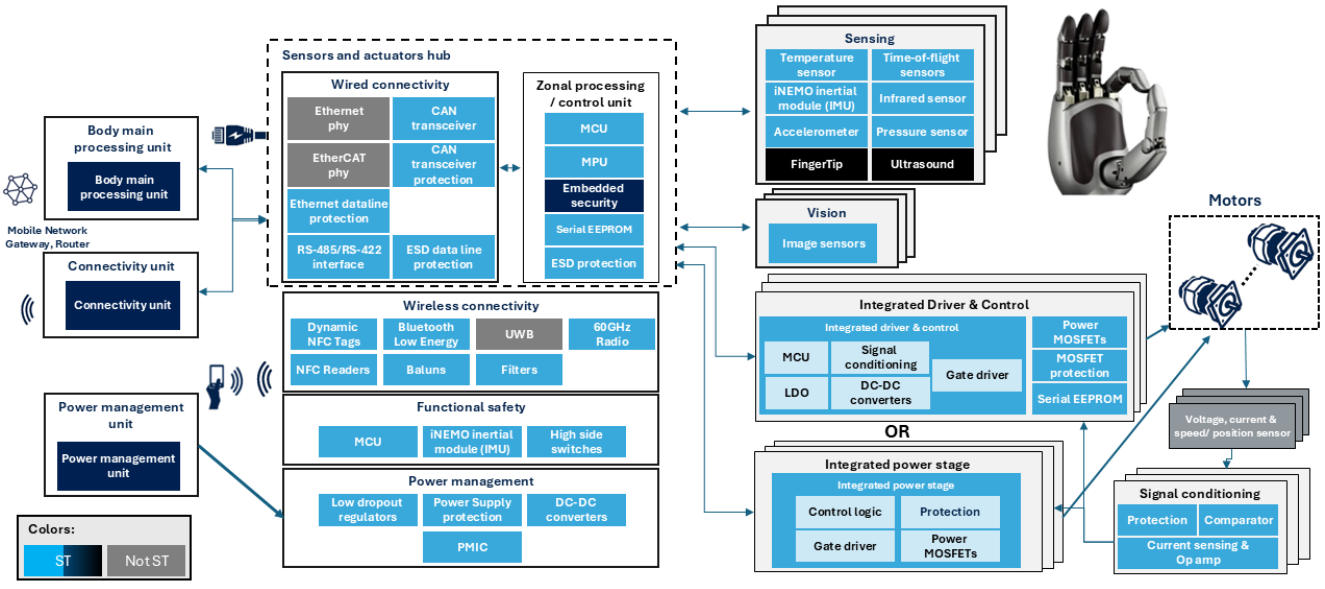
Design decision	Why it matters in humanoid robots	Architecture implication
Motor control and current acquisition	Following a desired torque trajectory depends on deterministic PWM generation and low-latency current feedback.	Codesign PWM, current sensing and ADC timing. Current designs are emerging from 12 or 24 V to 48 V, and 5 A or less per motor, with PWM often in the 20 to 50 kHz range. Next generation moving to the range 50 to 100 kHz PWM.
Position and contact feedback	Motor position alone may not represent joint position or fingertip interaction.	Use motor-side position feedback as the baseline. Depending on the mechanism, support one or two position or motion sensors per axis, typically through direct low-latency SPI interfaces, and add tactile, pressure or force-related sensing where needed.
Channel density and processing partition	Increasing DoF drives thermal density, wiring complexity, sensor traffic, and calibration effort.	Keep time-critical motor loops in the motor-control nodes. Use a separate bridge MCU for motor-node coordination, sensor aggregation, calibration, and arm communication. The current reference architecture illustrates scaling from 6-DoF to 16-DoF.
Communication, calibration, and fault handling	Dexterity depends on reliable sensing, persistent calibration, and predictable behavior during faults.	Use compact CAN FD or RS-485 links for robust communication inside the hand. T1S might reduce wiring in suitable multidrop configurations, but the number of motor control nodes might require multiple segments. The bridge MCU aggregates local networks, connects the hand to the arm controller, and supervises communication timeouts.

4 Design priorities for dexterous hands

An actuator system architecture describes how the control, communication, power, and sensing domains interact to realize motion control and force generation. Figure 4 illustrates an example of such an architecture.

The architecture decisions summarized in Table 4 must be adapted to the selected hand architecture, actuator placement, mechanical transmission, and sensing strategy. The following sections highlight the areas that most strongly influence dexterity, scalability, and system integration.

Figure 4. System block diagram for dexterous hands



4.1 Motor control and position feedback

Hand actuators frequently operate at low speed and under varying physical contact conditions. A smooth motion therefore depends on deterministic PWM generation, synchronized current acquisition, and precise position feedback.

One position sensor per controlled axis is sufficient in many implementations. A second sensor may be required to measure the position of a gearbox or tendon transmission, to compensate for backlash or elasticity, or to provide diagnostic diversity. Unlike many factory-automation servo drives, compact humanoid-hand actuators typically use short local connections to magnetic or other compact absolute position sensors. Direct serial peripheral interfaces (SPIs) on the motor-control MCU are therefore an important architectural requirement, particularly where data from two sensors per axis are acquired with low and predictable latency. MEMS motion sensors can provide independent information about the actual motion and orientation of selected finger segments or fingertips. When combined with encoder data, this feedback can help to verify commanded motion. It ensures that the motion results in the expected physical movement. It can also detect transmission errors or unexpected motion. Additionally, it improves observability when the motor-side position does not fully represent finger movement. This is particularly valuable in linkage and tendon-based hands. It helps when cameras lose sight. This loss can happen if the hand wraps around an object. It can also occur in smoke, water, or dirty environments where lenses get obscured.

4.2 Channel density, power, and thermal design

Many current hand architectures operate from 12 to 24 V supplies and low current per motor, allowing compact integrated motor-control solutions. Higher DoF counts and growing sensor density increase interest in 48 V-class power distribution. This system reduces conductor current and wiring losses. It also lessens the connector burden while providing extra power headroom. This trend also increases the importance of voltage margin, isolation strategy, driver selection, current-sensing design, and fault management. Some designs even use PWM frequencies between 50 kHz and 100 kHz when smaller motors, quieter operation, or higher efficiency are needed.

4.3 Contact sensing, calibration, and local processing

To execute precise tasks, the control system needs both mechanical and environmental feedback. It uses encoders for motor and joint positions, and pressure sensors to detect touch and force. Finger-segment or fingertip inertial measurement units (IMUs) can add direct motion and orientation information, particularly in palm-based or tendon-based hands, where linkage compliance, tendon stretch, friction, or hysteresis can make motor-side position an incomplete representation of actual finger movement.

IMU feedback can support dynamic finger-motion observation. It helps with transmission-state estimation. It also detects unexpected motion when visual sensing is unavailable or blocked. This capability is useful during enclosed grasping or in low-visibility environments. IMUs do not directly measure contact force and should complement, rather than replace, joint encoders and tactile or pressure sensing.

As the sensor count increases, local preprocessing can reduce the amount of raw data transmitted to the arm or central controller. In-sensor AI processing, such as finite state machine (FSM), machine learning core (MLC), and intelligent sensor processing unit (ISPU), can reduce microcontroller strain and power consumption. The palm controller can collect or filter touch and movement data. It leaves grasp planning, object understanding, and robot perception to the software of the customer.

Persistent memory helps to retain motor parameters, encoder offsets, transmission calibration, sensor calibration, and module identity. The architecture should distinguish among factory calibration, service calibration, module-replacement calibration, and compensation performed during operation.

4.4 Communication and fault handling

CAN FD and RS-485 are well suited for communication within the hand because they require a small number of MCU pins, support robust multidrop topologies, and can be implemented with compact transceivers. SPI and I3C remain appropriate for short, board-level connections to local sensors, or closely coupled devices.

The communication architecture should be selected according to the number of motor-control nodes, not the number of DoF alone. In a palm-based or tendon-based architecture, one motor-control MCU can control multiple motors. This setup keeps the number of network nodes relatively low. In an in-finger architecture, each motor may have a dedicated motor-control node, so a high-DoF hand can require multiple local network segments.

10BASE-T1S is an emerging option for reducing wiring through single-pair multidrop communication. However, one T1S segment supports a limited number of attached devices. A tendon-based architecture may remain within one segment when each motor-control MCU manages multiple motors. An in-finger architecture with one node per motor may require two or more T1S segments. The bridge MCU must then aggregate these segments and provide a separate CAN FD, EtherCAT, or TSN-capable connection to the arm controller.

Detachable hands and tools can use an ST60 short-range contactless data link combined with wireless power transfer to create a connector-less module interface. This can simplify hand or tool replacement, reduce connector wear, exposure to contamination, and increase operational flexibility. The bridge MCU provides a hand-level data interface, while deterministic motor control and immediate fault handling remain local to the hand. Any use of the contactless link for real-time commands or feedback must be validated against the required latency, availability, and fault-response targets.

Each motor-control node should detect:

- Local overcurrent
- Overvoltage
- Invalid sensing and overtemperature conditions
- Transition to a defined state

The bridge MCU should also supervise node communication, network timeouts, and hand-level coordination. Watchdogs, controlled-stop inputs, secure firmware updates, and protected calibration data can improve operation. However, present them as certified safety functions only with a documented system-level safety concept.

5 ST palm-based robot hand reference architectures

The following reference architectures are both palm-based. The compact 6-DoF example emphasizes centralized multichannel motor control, while the 16-DoF example shows how higher motor-channel density, richer sensing and optional local preprocessing influence the overall architecture.

Tendon-based and in-finger, direct-driven hands remain important design options. It is planned to include dedicated ST reference architectures in this application note later.

5.1 Compact 6-DoF palm-based architecture

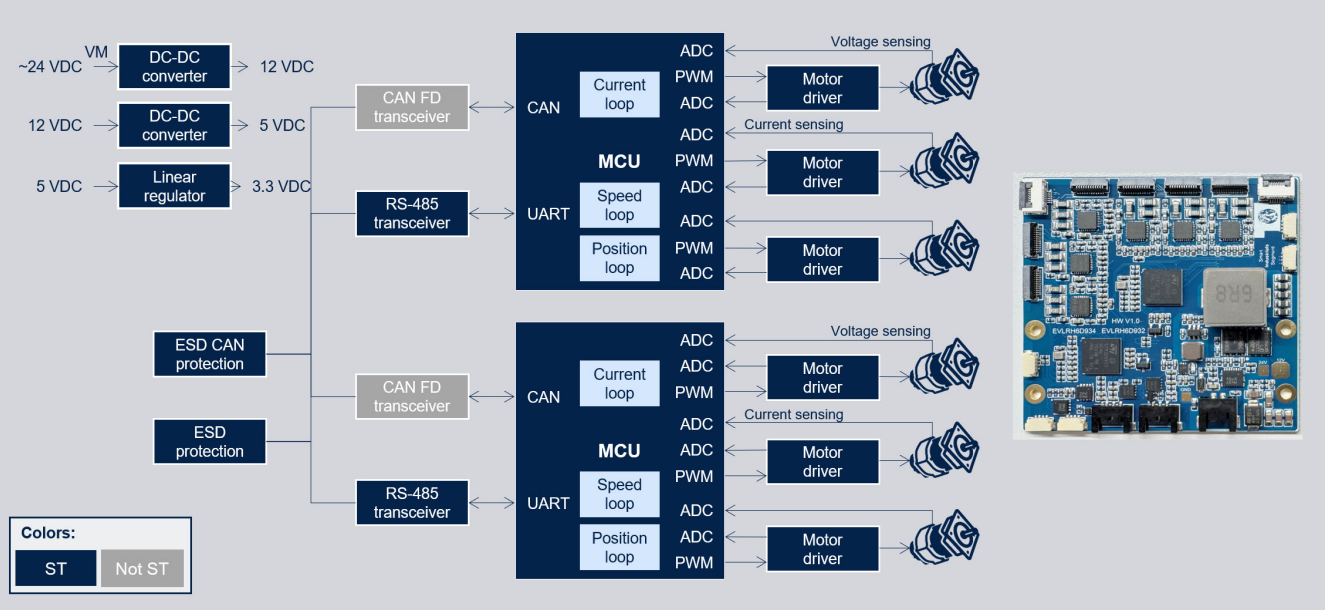
The compact 6-DoF architecture centralizes motor control, communication, calibration, and power management within the palm. It targets hands that require coordinated multiaxis motion while minimizing active electronics inside the fingers.

One or more STM32 motor-control MCUs manage the motor channels, synchronized current and position acquisition, and local diagnostics. A separate bridge MCU coordinates the motor-control channels and sensors, and connects the hand to the arm controller through CAN FD, Ethernet, or an external EtherCAT slave controller.

Features and benefits:

- Compact centralized architecture
- Reduced finger wiring and electronics
- Shared motor-control, power-management, and communication resources
- Reusable platform for coordinated hand motion

Figure 5. Compact 6-DoF palm-based hand reference architecture



5.2 Sensor-rich 16-DoF palm-based architecture

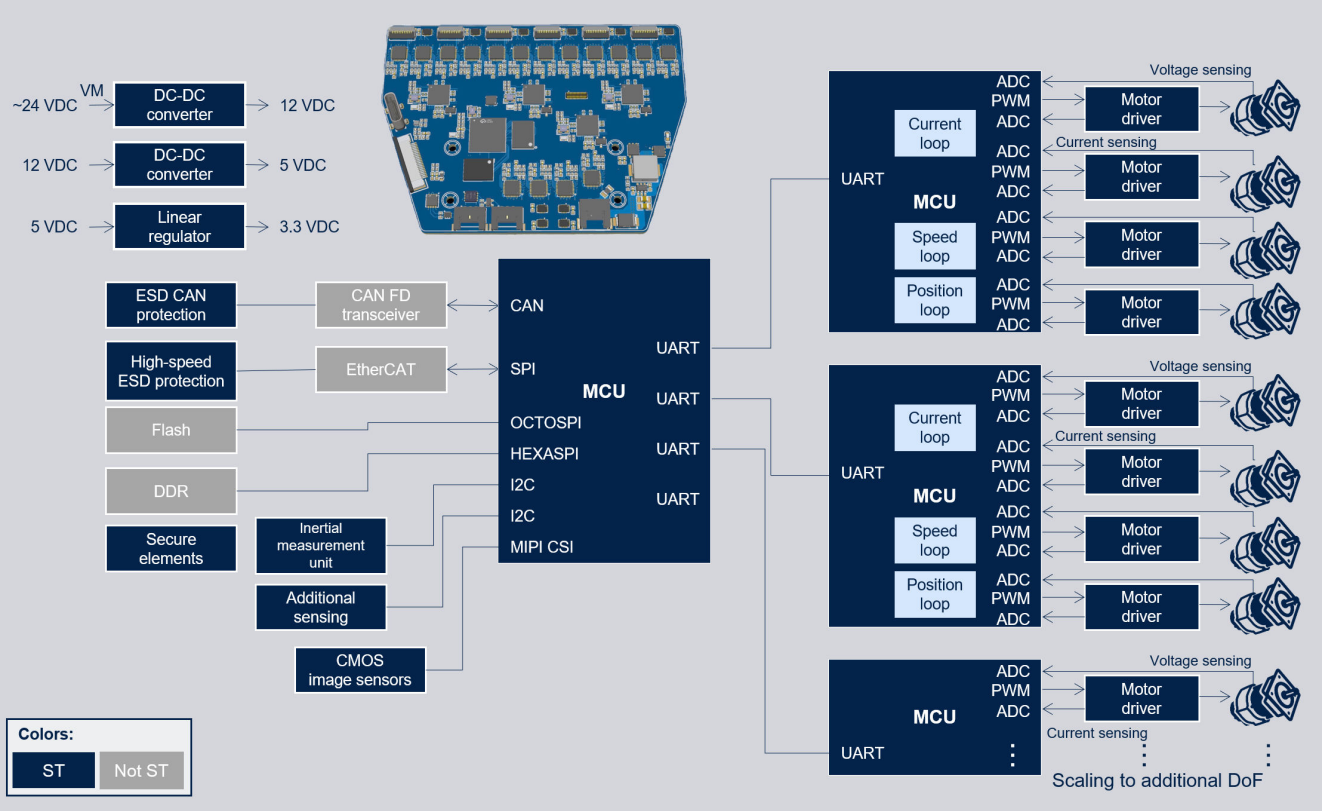
Scaling from 6-DoF to 16-DoF increases more than the motor count. It raises power and thermal density, PCB complexity, wiring, sensor traffic, calibration effort, and the number of possible fault conditions.

Useful dexterity also requires richer feedback. The architecture therefore evolves from a multiaxis motor controller into a sensor-actuator subsystem that combines actuation with position, tactile, pressure, and inertial sensing

In addition to palm-level motion sensing, the architecture can integrate IMUs on selected finger segments or fingertips. These sensors measure motion and orientation after the mechanical transmission and complement motor-side encoders, especially in linkage-driven or tendon-driven designs. Tactile or pressure sensors provide contact feedback for grasp adaptation. A compact camera module, combined with an STM32N6 series edge AI processor, performs local object detection, gesture recognition, or preprocessing for hand-eye calibration. Then, it forwards selected information to the robot arm or central robot controller.

Local processing does not replace the robot main motion-planning or perception system. Its value is to reduce raw sensor traffic, shorten the response path for hand-specific functions and provide a reusable interface between perception, grasp coordination, and multiaxis actuation.

Figure 6. Sensor-rich 16-DoF palm-based hand reference architecture with local perception



The 16-DoF implementation extends the common palm-based control platform with higher motor-channel density, richer sensing and optional local perception. This supports more adaptive grasp behavior while reducing the amount of time-critical motor and raw sensor data transmitted to the higher-level robot controller.

Table 5. Selected ST products for the sensor-rich 16-DoF palm-based hand architecture

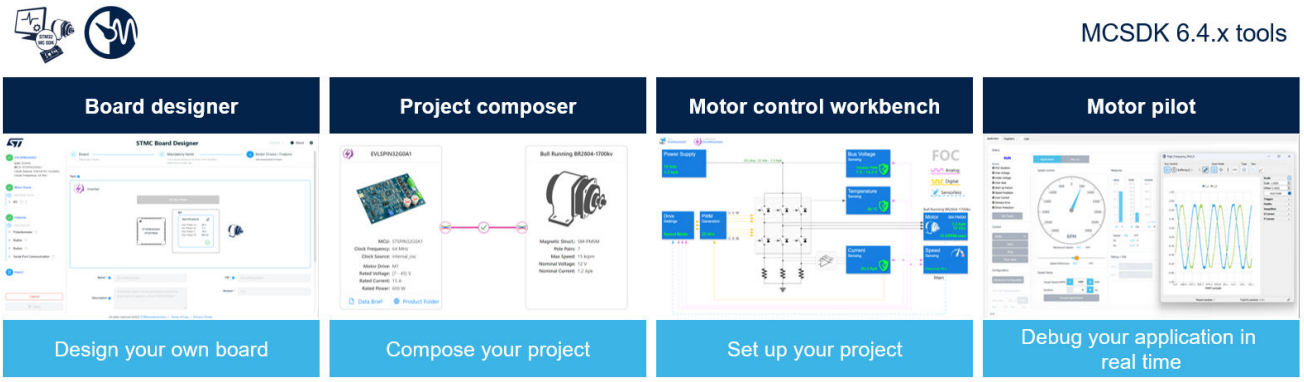
Function block	ST building block	Purpose
Motor-control platform	STM32G4 series(1)	Deterministic FOC execution, PWM generation, synchronized current acquisition, encoder feedback, and local motor-node protection
Bridge controller	STM32H5 series STM32H7 series	Coordinates motor-control nodes, aggregates hand sensors and calibration data, manages diagnostics, and connects the hand to the arm controller
Perception processing	STM32N657L0HxQ(2)	Local preprocessing of selected image, tactile or high-volume sensor data, separate from deterministic motor control and bridge functions
Compact motor driver	STSPIN932(2) STSPIN934(1)(3)	Integrated three-phase drive, current sensing and local protection for space-constrained palm electronics
Time-of-Flight	VL53L4, VL53L5, VL53L8, VL53L9	Object detection, positioning, alignment, and grasping
CMOS image sensors	VD55G1, VD16GZ, VDx943, VD1940	Visual guidance for precise grasping
Capacitive or pressure sensing for tactile	FST3CA60Y FST3CA67Y FST3CA84Y ILPS22QS ILPS28QSW	Touch and contact sensing for grasp control
Motion sensing	ISM330DHCX ISM330IS LSM6DSV16X(2)	Palm, finger-segment, or fingertip motion feedback Transmission-state observation Vibration monitoring and diagnostics
Optional sensing	Magnetometer: IIS2MDC Magnetic switch: NMH1000 Biosensor: ST1VAFE3BX	Alignment and organic material detection
Local communication	ST4E1240IQT(1)(2) RS-485 CAN FD protection	Robust communication at: • Finger level • Palm level • Arm level
Contactless detachable-module interface	ST60 series	Short-range data connectivity for detachable hands or tools, with optional wireless power transfer
Secure element	STSAFE-A120	Device authentication, secure attachment, system integrity, including provisioning services
NFC reader	ST25R210 ST25R300	Access management, accessory recognition, and authentication
Local power management	DCP3601NDMR(1)(2) L3751PUR(1) LD39150PUR33R(2) LDL212PV33R(1) DC-DC and LDO devices	Regulated rails for control, sensing, and communication

1. *Used in the application as shown in Figure 6*
2. *Used in the application as shown in Figure 7*
3. *In development, engineering samples are available*

6 Software enablement for dexterous hand control

Robotic-hand developers use proprietary software, open motor-control platforms, and vendor-supported software packages according to their existing architecture and development stage. The motor control software development kit (X-CUBE-MCSDK) is an ST motor-control software expansion package for precise and efficient control of three-phase motors on supported STM32 devices. In a robotic hand, the main value comes from running advanced control algorithms. High-performance STM32 microcontrollers provide enhanced processing capability. They also offer analog peripherals and timing resources. This setup allows for sophisticated motor control functions. It ensures high dynamic performance and tight control loop execution. Some robotics developers use other platforms, such as ODrive. The appropriate approach depends on the existing software architecture, required feedback interfaces, performance targets, and validation needs.

Figure 7. Key components of X-CUBE-MCSDK version 6



For robotic hands, the requirements are often even more demanding than for larger joints in terms of compactness, precision, responsiveness, and smoothness of motion. Each finger or thumb actuator must deliver accurate torque control in a very small form factor, often with strict limits on weight, power consumption, and thermal dissipation. X-CUBE-MCSDK supports these constraints by providing a flexible control framework for three-phase motors that can be adapted to compact actuator architectures while maintaining reliable and efficient operation.

From an algorithmic point of view, X-CUBE-MCSDK supports several control strategies depending on the motor type, sensing method, and application constraints. The package can implement six-step commutation, although this approach is generally less relevant for robotic hands, where smooth motion and fine control are usually required. The core strength of the package is field-oriented control (FOC), which can be deployed in both sensored and sensor-less configurations. Sensored FOC can use incremental encoders or *Hall* sensors, while sensor-less operation can rely on estimators such as the *Luenberger* observer or more advanced approaches like HSO and ZeST.

This flexibility allows designers to select the most appropriate control scheme for each finger or gripping actuator, balancing cost, performance, and mechanical integration constraints.

A key advantage of X-CUBE-MCSDK is that it is not a standalone tool. It is fully integrated into the ST ecosystem and works together with STM32CubeMX, which generates the project configuration and the underlying libraries. This significantly simplifies system integration with other ST products and helps streamline development from initial setup to final firmware generation. For robotic hand applications, where motor control must often be combined with tactile sensing, communication interfaces, safety logic, and higher-level grasping algorithms, this ecosystem-based approach reduces integration effort and improves consistency across the full embedded stack.

ST MEMS sensors can complement the motor-control software through embedded processing and STM32 motion-sensor libraries. Machine learning core, finite state machine, or ISPU-enabled sensors can execute functions such as motion and shock detection, vibration monitoring, and activity recognition close to the sensing point. STM32 software libraries can additionally support sensor calibration, sensor fusion, and real-time motion detection. This processing can reduce bridge-MCU and network load while providing finger-motion or palm-motion information for state estimation, diagnostics, and transmission monitoring. These functions complement, rather than replace, encoder and tactile feedback. The support must be verified for the selected sensor, MCU, and released software package.

The package also includes a motor-control protocol interface, which communicates with the MCU over UART and enables real-time debugging of the control algorithm. Through this interface, users can monitor live variables, observe internal control states, and tune parameters during operation. This is especially valuable during the development of robotic hands, where optimizing low-speed behavior, startup performance, and grip response is critical to achieving natural and stable manipulation.

At the same time, it is important to define the current boundaries of the solution. X-CUBE-MCSDK does not yet support absolute encoders, which can be a significant limitation for robotic hand systems that require immediate position knowledge after power-up or after a fault recovery. However, the package does provide the fundamental motor-control functions, such as FOC, position detection, and standard control loops, but it does not include more advanced application-specific features such as moment-of-inertia compensation, cogging torque compensation, torque vectoring, friction compensation, or calibration management. These capabilities must still be implemented at the application level by the user, depending on the mechanical design and performance targets of the robotic hand.

From the safety perspective, X-CUBE-MCSDK includes a broad set of protections and fault-handling mechanisms that help improve operational robustness. However, it does not by itself provide certification-oriented safety frameworks such as class B or higher. For applications that require safety-compliant software infrastructure, ST offers dedicated safety packages, including X-CUBE-CLASSB, which provide sets of safety functions aligned with specific STM32 microcontrollers. This separation allows developers to combine high-performance motor control with the appropriate safety layer when the target application demands it.

Overall, X-CUBE-MCSDK is best understood as a powerful motor-control foundation for robotic hand systems, particularly when used with high-end STM32 devices. It provides the control algorithms, integration tools, and debugging capabilities needed to build advanced three-phase motor drivers, while leaving room for application-specific optimization and higher-level robotic functions to be implemented above the motor-control layer. For robotic hands, this makes it a strong enabler for compact, precise, and scalable actuator design.

7 Conclusion

Dexterous humanoid hands are distributed sensor-actuator systems in which actuation, sensing, calibration, communication and local processing must be designed together. Palm-based, tendon-based and in-finger, direct-driven architectures each create different tradeoffs in moving mass, integration density, thermal design and feedback strategy.

The palm-based 6-DoF and 16-DoF reference architectures, shown respectively in Section 5.1 and Section 5.2, demonstrate how a common control platform can scale from compact multi-axis control to sensor-rich dexterous manipulation while maintaining a clear separation between motor control, palm-level coordination and optional perception processing.

Revision history

Table 6. **Document revision history**

Date	Version	Changes
13-Aug-2026	1	Initial release.

Contents

List of tables

List of figures