



# Software package for efficient use of a robotic anthropomorphic hand

Marina Pujol-Closa  
*Inst. of Industrial and Control Eng.*  
*Universitat Politècnica de Catalunya*  
Barcelona, Spain  
marina.pujol.closa@upc.edu

Leopold Palomo-Avellaneda   
*Inst. of Industrial and Control Eng.*  
*Universitat Politècnica de Catalunya*  
Barcelona, Spain  
leopold.palomo@upc.edu

Raúl Suárez   
*Inst. of Industrial and Control Eng.*  
*Universitat Politècnica de Catalunya*  
Barcelona, Spain  
raul.suarez@upc.edu

**Abstract**—The aim of this work is the development of a complete software package to exploit a robotic anthropomorphic hand, such as the Allegro Robotic Hand, allowing to easily perform accurate grasps. It includes the Inverse and the Forward Kinematics and allows storing different configurations as well as to perform and store predefined grasps while controlling the movement speed. The software presented in this document allows the easy use of the Allegro Robotic Hand, enhancing its capabilities and allowing further research on grasp theory by facilitating the experimental implementation. The software is designed to work with all the Allegro Hand versions despite their differences. It is an improved alternative to the manufacturer’s version, offering a powerful framework for grasping. Besides, it has been designed to easily adapt to other robotic anthropomorphic hands.

**Index Terms**—Robotics hands, Allegro Hand, ROS.

## I. INTRODUCTION

Anthropomorphic robotic hands are essential tools in advanced robotics, offering high dexterity and versatility for a wide range of applications [1]. The synergy between hardware and software is crucial for their optimal performance and usability [2]. The focus of this work is the development of a software package for the efficient use of robotic anthropomorphic hands, specifically the Allegro Hand (AH) by Wonik Robotics, although it is designed to easily adapt with other robotic anthropomorphic hands. The official AH software manages communications, has a Gravity Compensator, a Proportional-Derivative (PD) controller and four predefined grasps.

Additionally, it provides through partial Forward Kinematics the position of the fingertips (but not the orientation). However, the proprietary nature of the BHand library, which is distributed only in a binary format, limits the user capability to fully customize and enhance the system.

This paper introduces a new open-source C++ software that addresses these limitations, offering a solution that enables communication and control of all the Allegro Hand versions. It includes full implementation of both Forward and Inverse Kinematics (FK and IK), allowing users to move the fingertips to precise positions and orientations. It also supports predefined and customizable hand configurations and grasp motions. Additionally, it contains different control modes,

such as Torque, Gravity Compensator, PD and Proportional-Integral-Derivative (PID). This work is the result of an effort to provide useful and open tools to facilitate the research on grasping and manipulation.

By providing a comprehensive and adaptable software package, this contribution significantly enhances the application of the AH while simplifying its usability. It facilitates more accurate and versatile grasping capabilities, enables the storage and execution of predefined configurations and supports research in grasp theory by simplifying the implementation of theoretical models. Moreover, the software has been designed to be easily adapted to other robotic anthropomorphic hands, making it a versatile tool for advanced robotic manipulation research and applications. To enhance integration with other robots and sensors, a ROS Noetic and Humble packages have also been developed. All the developed software is available at [https://sir.upc.edu/projects/allegro\\_hand/](https://sir.upc.edu/projects/allegro_hand/).

## II. THE ALLEGRO HAND

### A. The Allegro Hand hardware

The Allegro Hand, illustrated in Fig. 1, has four fingers, each with four independent current-controlled joints; abduction (joint 0), proximal (joint 1), middle (joint 2), and distal (joint 3), totaling 16 Degrees of Freedom (DoF). The main differences compared to a human hand are:

- The distal and middle joints are not coupled, they can be flexed independently, providing a better range of action.
- The abduction joint rotates along an axis aligned with the extended finger, whereas the abduction of a human hand rotates along an axis orthogonal to the palm.

### B. The Allegro Hand communication protocols

The AH is commanded by sending a message with the joint torques of each finger and reading another message with the values of the finger joint positions. Thus, acquiring/setting the configuration of the whole hand requires four messages. The communication is periodic and each received/sent message contains the information of the four joints of a finger. Each message has an identifier indicating the related finger and the messages are always sent in the same order: Index, Middle, Annular and Thumb.

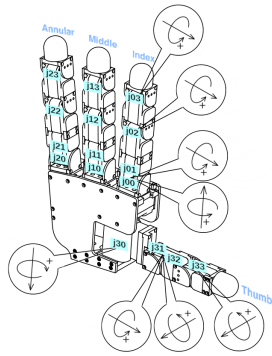


Fig. 1. Schematic of the Allegro Hand showing the labeled joints and their rotation axes.

There exist four different versions of the AH, all sharing the same physical dimensions and kinematics. The primary differences among these versions lie on the communication protocols and on the characteristics of the servos. Up to version 3, the communication protocol is almost the same, while in version 4 it is different. The basic difference is the message length. Each message has an Identifier and the Data, that until version 3 are:

- Message Identifier: 4 bytes (26 bits: Command Identifier, 3 bits: Source Identifier and 3 bits: Destination Identifier).
- Data: 8 bytes (the messages related to reading joint states and sending the desired torques assign 16 bits per joint).

whereas the structure of version 4 messages is:

- Arbitration identifier: 11 bits (9 bits: Message Identifier and 2 bits: Device Identifier)
- Data: up to 8 bytes, with the length depending on the type of messages; those related to reading joint states and sending the desired torques assign 16 bits per joint.

The reduction of the message length is an improvement, but the process of creating and reading the messages is also different depending on the version. For each version, the manufacturer provides a different way to convert the current of the servos into torques. In versions 1, 2 and 3 there exists an encoder offset, an encoder direction and a motor direction. These values are different for each of the 16 joints and for each manufactured hand, the relationship between the angles and the received data being:

$$\text{joint}[\text{rad}] = (\text{encDirection} \cdot \text{data} - 32768.0 - \text{encOffset}) \cdot \left( \frac{333.3}{65536.0} \right) \cdot \left( \frac{\pi}{180.0} \right) \quad (1)$$

whereas in version 4 there is no encoder offset and all the encoder directions and motor directions are well adjusted to 1 for all the manufactured hands, the relationship being:

$$\text{joint}[\text{rad}] = \text{data} \cdot \left( \frac{333.3}{65536.0} \right) \cdot \left( \frac{\pi}{180.0} \right) \quad (2)$$

Another significant difference is that the empirically read values of the joint limits are not the same for all the versions, nor for all the hands within a version, except for version 4, in

which the values are consistent. Furthermore, the read limits are not consistent with the limits of the Unified Robot Description Format (URDF) model provided by the manufacturer. This issue has been solved in the developed software, to ensure the same performance and compatibility across all versions and hands, using the real full available range.

### C. The official Allegro Hand software

The official AH software distributed by Wonik Robotics [3] consists of several packages: a shared library called BHand and one version-specific package for GNU/Linux and for Windows. Additionally, a ROS 1 (Kinetic) package for GNU/Linux is provided. The BHand library is distributed only in binary format, with the necessary header files but not the full source code. It offers two controllers: the Gravity Compensator (without considering the hand orientation) and the PD, and it also has partial Forward Kinematics (without orientation). It also embeds four types of grasps: using 3 fingers, using 4 fingers, pinch with index and thumb and pinch with middle and thumb.

The version-specific package has an instance of the BHand library, manages communications and includes some predefined configurations. A unique thread continuously reads the messages received from the hand. Upon receiving a message with the current joint positions, the desired torque is calculated using the BHand library; first it updates the current and desired joint positions and, then, executes the control algorithm to obtain the desired torques.

The choice of the control algorithm significantly impacts the hand performance. While the official BHand library utilizes a PD controller, other controllers like PID offer several advantages, such as correcting steady-state errors more effectively, and are better suited for grasping objects and holding a position.

Although the manufacturer allows to read the current fingertip positions through the FK, it does not provide the fingertip orientation. The full FK is crucial to work with the hand in Cartesian coordinates. Moreover, there is no built-in support for IK, which is a useful tool to control the hand by sending fingertip positions (X, Y, Z coordinates) and orientations (Euler angles), instead of joint torques. This gap is addressed in the developed software.

## III. THE DEVELOPED SOFTWARE PACKAGE

The official software is useful but it lacks many important features that limits the ability to fully customize and enhance the system, thus, a new open-source and more complete package is proposed here.

The presented Allegro Hand Library implemented in C++ has been developed to simplify the use of the hand, while enhancing its functionalities. It handles communications, complete FK and IK, several control modes and has both predefined and customizable configurations and grasp modes. It allows decoupling the drivers (communications and control) and the framework, easing the integration with other frameworks (e.g. ROS 1, ROS 2, Orocos, Matlab).

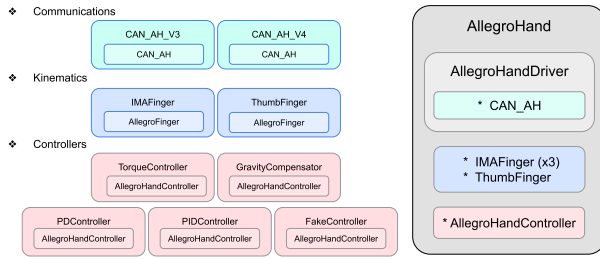


Fig. 2. Structure of the classes of the developed software package.

## A. Classes and functions

The code structure is divided into five classes (see Fig. 2):

1) *CAN\_AH*: It enables communication with the AH. It allows to start and stop communications, generates messages to send torques and parses the received messages to get the current configuration.

2) *AllegroHandController*: The software has five different control modes: Torque, Gravity Compensator, PD, PID and Fake controller. (each controller being a subclass of the *AllegroHandController* class):

- *Torque*. Sets and maintains the desired torques.
- *GravityCompensator*. Calculates the needed torques to counteract the effects of gravity using the Kinematics and Dynamics Library (KDL).
- *JointPositionPD*. Uses a PD control to achieve desired joint positions, considering gravity effects.
- *JointPositionPID*. Uses a PID control to achieve desired joint positions, considering gravity effects.
- *Fake*. Used only for simulation, computes the gravity compensation torques and updates the virtual hand state.

3) *AllegroFinger*: It has two child classes, one for the Index, Middle and Annular (IMA) fingers and another for the Thumb. The IMA fingers structure is the same, whereas the Thumb kinematics is different. This class contains the implementation of FK and a closed form of the IK of the IMA and Thumb fingers. The implementation allows to move the fingertips to the desired positions (X, Y, Z coordinates) and orientations (Euler angles) through the use of the IK and read the current poses of each fingertip through FK.

4) *AllegroHandDriver*: It is a basic driver to communicate with the hand. It reads the hand configuration file and allows reading the current joint positions and setting specific torques. It contains an instance of the *CAN\_AH* class.

5) *AllegroHand*: The *AllegroHand* class is the most crucial, since it is a derived class from the *AllegroHandDriver* and embeds instances of the other classes. When initialized it reads a configuration file, the URDF and predefined joint configurations and grasp modes (see Fig. 3). It can run both, a real Allegro Hand or a simulated one. It instantiates a *AllegroController* class (Torque or Fake for real or simulated, respectively) and four instances of the kinematic classes (one per finger). In both cases it contains a thread executing the main -and only- loop, *controlLoop* or *controlLoopFake*. When working with a physical hand, it initializes an instance of

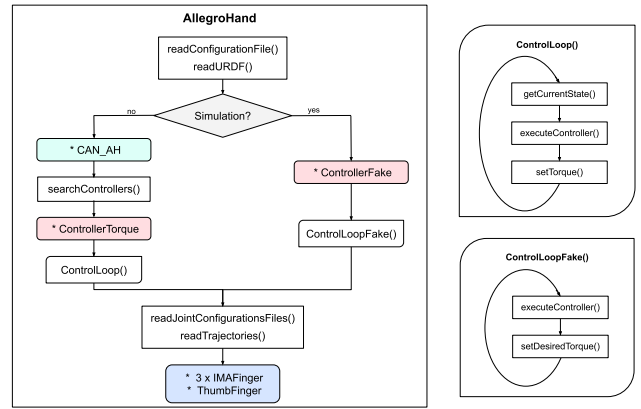


Fig. 3. Structure of the *AllegroHand* class.

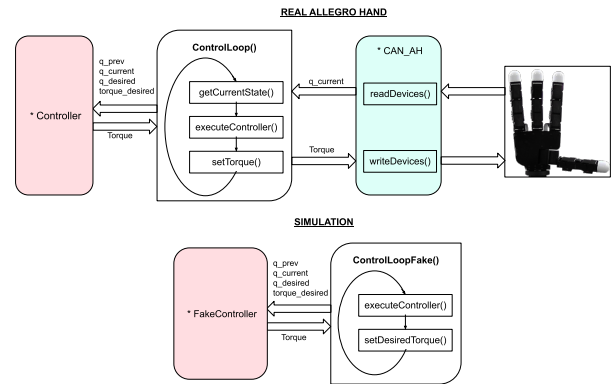


Fig. 4. Main control loop for the real and the simulation case.

the *CAN\_AH* class for communication and finds the available controllers. The *controlLoop* reads current joint configurations, executes the current controller and sends the desired torques to the Allegro Hand. The *controlLoopFake* executes the fake controller, ensuring the desired torques are within the limits of the simulated hand (see Fig. 4).

The *AllegroHand* key functions include:

- *Control algorithms*. It allows users to obtain a list of the available controllers, set the desired controller, get the current controller and obtain joint position errors. It can also get and set the hand gravity vector, used to calculate the necessary torque to compensate the gravity effect, considering the hand orientation. This feature is important when the hand is installed on a robot arm whose movement affects the hand gravity vector. When switching the controller, the previous controller instance is deleted, ensuring efficient memory management.
- *Hand configuration*. It allows to obtain the current joint positions and set the desired ones. It can read YAML files with predefined configurations, list all saved configurations, temporarily add and delete configurations from the list, permanently store a configuration into the desired YAML file and move the hand to a new or a predefined configuration, among other functionalities.
- *Grasps modes*. It allows to move the hand to a predefined grasp configuration along a given trajectory. Additionally,

it can read and store trajectories to perform different grasp types from YAML files.

- *Forward and Inverse Kinematics.* These functions compute the FK and the IK of the hand. Moreover, there is a function to obtain the Jacobian, which uses the kinematic chain built from the URDF model using the KDL library.
- *Joint limits.* A program, *calculate\_joint\_limits*, has been developed to store in a YAML file the read minimum and maximum real joint values. The problem of the discrepancies in the joint limits between the manufactured hands and the official URDF limit values has been solved by changing the URDF limits using the empirical read values from version 4, which exhibit consistency across manufactured hands. For earlier AH versions, the program can also adjust the *encoderOffset* values in the hand configuration file to minimize limit differences and center the range. The developed software accounts for both the URDF limits and the real limits from the YAML file, including a safety margin. Functions are provided to obtain joint limits, set a safety margin (by a general value, percentage, or specific value for each joint), and choose whether to apply the safety margin.

The software allows to test the AH driver and the AH library through a set of example files located in the *examples* folder. To initialize the AH, it is required the hand configuration file, located in the *conf* folder among other useful YAML files that contain information of the controller parameters, the URDFs, a set of predefined joint positions and predefined grasps and the actual joint limits of each tested hand.

### B. Poses and Configurations

The developed software allows to move the hand to a desired configuration and to store the current configuration. During the hand movement, it checks whether the Gravity Compensator or the Torque controller are being used, if so, it switches to a PID controller. Additionally, users can manually adjust the hand to a desired configuration and store it using the Gravity Compensator controller.

The analogous physical configurations for the right and the left hands do not correspond to the same joint values due to differences in joints 0 of the IMA fingers. These discrepancies are accounted for, and, upon saving a hand configuration, the corresponding appropriated configuration for the opposite hand is calculated and stored.

### C. Grasping Modes

Several grasping types are pre-programmed as list of configurations in a YAML file. Whenever a hand trajectory is saved, the analogous trajectory for the opposite hand is also stored. Grasps can be stored in two ways. The first way is as follows, the software allows to move the hand to a configuration at a desired speed, ensuring all fingers reach the final position simultaneously. It calculates the difference between current and final desired joint values, using the largest difference to determine the number of intervals needed to reach the final hand configuration at the desired speed, considering

the sample period. Then it computes the differential values for each joint, creating a list of interpolated configurations between the initial and final configurations. The second way to generate a grasp is by recording the positions of hand being manually manipulated.

A selection of grasps that allow in-hand manipulation, specifically all types of prismatic precision grasp [4], have been already implemented.

### D. Validation

The C++ software, the ROS Noetic and the ROS Humble packages have been tested with various Allegro Hands, including models R283 and R284 from version 4, as well as SAH020AL011 and SAH020BR014 from version 2. Videos illustrating the use of the developed C++ and ROS Humble packages are available in [https://sir.upc.edu/projects/allegro\\_hand](https://sir.upc.edu/projects/allegro_hand).

## IV. CONCLUSIONS AND FUTURE WORK

This paper presents the implementation of a fully developed C++ software package to improve and facilitate the use of robotics hands, and in particular, the AH. It provides crucial functionalities and simplifies integration with other frameworks, laying a solid foundation for future work in grasping research. It also facilitates the addition of new functionalities within the software, such as an specialized controller to enhance grasping capabilities, which is currently under development. Furthermore, the software is designed to be easily integrated with packages containing customized controllers, thus eliminating the need to allocate the controller within the package.

To promote integration with other manipulator robots and sensors, packages in ROS 1 (Noetic) and ROS 2 (Humble), that use the C++ software, have been created. Currently, the integration of the AH with robots, like MADAR [5], is being developed using the ROS framework. Additionally, a C++ driver for the Weiss tactile sensors and its ROS framework is also being developed.

Future work consists in connecting the arms and the sensors with the AH in order to implement force-controlled grasps, thereby fully exploiting the hand functionalities. Additionally, the software is designed to be compatible with other robotic hands with similar configurations, such as the LeapHand [6].

## REFERENCES

- [1] I. Llop-Harillo, J. Iserte, and A. Pérez-González, "Benchmarking anthropomorphic hands through grasping simulations," 02 2022.
- [2] R. Suárez and P. Grosch, "Dexterous robotic hand ma-i, software and hardware architecture," in *Intelligent Manipulation and Grasping International Conference, IMG'04*, Jul 2004, pp. 91–96.
- [3] W. Robotics. Programming guides — allegro hand. [Online]. Available: <https://www.allegrohand.com/programming-guides>
- [4] M. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *Robotics and Automation, IEEE Transactions on*, vol. 5, pp. 269 – 279, 07 1989.
- [5] R. Suárez, L. Palomo-Avellaneda, J. Martínez, D. Clos, and N. García, "Manipulador móvil, brazo y diestro con nuevas ruedas omnidireccionales," *Revista Iberoamericana de Automática e Informática industrial*, vol. 17, no. 1, pp. 10–21, 2020.
- [6] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," *Robotics: Science and Systems (RSS)*, 2023.