Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina 2026

External force-torque sensor feedback on the ABB GoFa collaborative robot

Łukawski, B.*, Olano Díaz, A., González Villasante, A., Oña, E. D., Victores, J. G., Jardón, A.

RoboticsLab, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911, Leganés, Spain.

Resumen

Este trabajo presenta la integración de un sistema de adquisición original y de código abierto, desarrollado para un sensor fuerza-par JR3 de seis ejes gestionado por un microcontrolador Mbed, y el brazo manipulador colaborativo CRB 15000-5 "GoFa" de ABB. La característica Externally Guided Motion (EGM) del fabricante es aprovechada para producir comandos de alta frecuencia mediante un dispositivo externo y la realimentación del sensor. Una arquitectura distribuida de componentes software es orquestrada por el middleware ROS 2 usando paquetes y librerías de Python. El sistema ha sido validado en simulación con RobotStudio y sobre el robot real mediante tres aplicaciones de alto nivel que emplean la realimentación de fuerza: control de admitancia (guiado kinestésico), control de impedancia y ejecución de trayectorias predefinidas con aplicación de presión constante. Se muestra un caso de uso de la última aplicación para realizar dibujos con el robot. Futuras extensiones de este sistema pueden ser aplicadas en escenarios que involucren teleoperación y realimentación háptica.

Palabras clave: Sistemas embebidos y aplicaciones de control por computador, Sistemas de control de movimiento, Percepción y sensorización, Tecnologías robóticas, Manipuladores robóticos

Abstract

This work presents the integration of an original and open-source acquisition system, developed for a six-axis JR3 force-torque sensor managed through an Mbed microcontroller, with the ABB CRB 15000-5 "GoFa" collaborative manipulator arm. The vendor-specific Externally Guided Motion (EGM) feature is leveraged to achieve high-frequency motion commands originating from an external control device and sensor feedback. A distributed architecture of software components is orchestrated by the ROS 2 middleware through Python packages and libraries. The setup has been validated in simulation via RobotStudio and the real robot through three high-level applications augmented with force feedback: admittance control (kinesthetic guidance), impedance control, and predefined trajectory execution with fixed pressure. A drawing task performed by the robot is presented as a use case of the latter application. Future extensions of this system can be applied to teleoperation scenarios and haptic feedback.

Keywords: Embedded computer control systems and applications, Motion control systems, Perception and sensing, Robotics technology, Robots manipulators

1. Introduction

Stiff position controllers enable industrial robots to follow a path with great accuracy and speed. However, compliant behavior proves best for scenarios in which readiness for contact with the environment is assumed, for instance, due to expected or unexpected obstacles along said path. Force feedback, achieved via internal or external sensing modules, allows robotic systems to adapt either online or offline trajectories to live and changing external conditions. In addition to solving new problems, damage to the work object, the tool, or the robot itself can be prevented altogether with compliant controllers.

This work iterates over previous developments which explored ABB's solutions for remote command generation, and also introduces the low-level acquisition and treatment of force-torque sensor data from a specific vendor, paving the way for the integration in an ABB robot. Even though it is of the collaborative kind ("cobot") and contact forces on the tip can be estimated via intrinsic torque measurements per axis, directly obtaining forces and moments via an external sensor on the tool flange was deemed best for developing new applications.

Recently explored educational contexts could be targeted by the tools described next (Łukawski et al., 2026).

*Corresponding author: blukawsk@ing.uc3m.es

This document is organized as follows. Section 2 summarizes similar and related works. Section 3 lists the hardware components involved in the setup. Section 4 describes the proposed architecture, whereas its experimental validation is laid out in Section 5. Finally, conclusions are drawn in Section 6.

2. Background

Physical interaction of robots with the environment, including surrounding people, has been extensively studied, even more so since the advent of collaborative robots, which share their workspace with human operators. Real-time collision detection algorithms, along with control strategies that promote safe robot reaction, are widely adopted in numerous robotic research-oriented and commercial systems to tackle this problem (Haddadin et al., 2017).

In the context of this work and the RoboticsLab research team it is set in, the TEO humanoid robot has long served as a versatile development platform (Pérez Martínez et al., 2010). It features four JR3 force-torque sensors in two models: smaller ones mounted on the wrists, and larger ones mounted on the knees. The former model has been used in the present work.

Early studies of TEO's interaction with its environment stem from locomotion-oriented tasks. During bipedal gait, a whole-body balance controller based on a simplified humanoid model was designed to avoid overturning, introducing the definition of several stability zones (Monje et al., 2018).

Focusing on manipulation tasks, TEO was also devised as a robot waiter that serves beverages on a tray it is holding with one hand. An extension of the whole-body controller was proposed to leverage the force-torque sensor located at the wrist (Garcia-Haro et al., 2020). Besides, sensor fusion of force-torque measurements and distance information acquired by a depth sensor was investigated in terms of a dual-arm lifting task. Wrist sensors played a role in the adequate grabbing of boxes and similar objects with both TEO's hands. Reinforcement learning techniques helped characterize the elasticity of grabbed objects and enabled the identification of a small subset of those (Hernandez-Vicen et al., 2024).

This work has also a strong focus on ABB robots and a specific commanding technique: the Externally Guided Motion (EGM) controller feature. Its language-agnostic nature enables its inclusion in varied contexts depending on the task and requirements at hand. For instance, the "EGM Research Interface (EGMRI)" library was used while interfacing with an ABB YuMi robot through the Julia programming language to apply motion correction (Bagge Carlson and Haage, 2017). On a similar note, a Simulink package was developed to integrate motion control and path correction with force feedback from an external force-torque sensor mounted on an ABB IRB 2400 industrial manipulator. It showcases the need of an supplementary controller due to incompatibilities between EGM and the "Force Control" feature, which uses joint torque measurements to estimate forces on the tip (Obal and Gierlak, 2021).

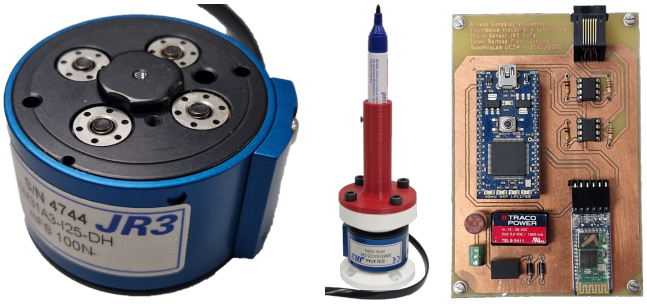
Lastly, a range of EGM-based applications have been recently developed at RoboticsLab with the aim of exploring its suitability in educational contexts, seeking the introduction of force control in industrial robotics laboratory sessions for undergraduate and graduate students (Łukawski et al., 2025a).

3. Materials

3.1. Force-controlled drawing tools

Figure 1 depicts a set of tools involved in the drawing scenario. At the core of this setup lies a JR3 6-axis force-torque sensor (model 50M31A3-I25-DH 100N5, depicted in Figure 1(a)). In order to mount it on the robot, due to incompatible diameters, a pair of adapters have been 3D-printed to interface between the robot's mounting flange and the sensor on one side, and the sensor and a tool on the other one. The selected tool is a pen holder with a marker inserted in it (Figure 1(b)).

Data acquisition from the sensor is achieved with an Arm Mbed NXP LPC1768 microcontroller, receiving data and clock serial streams through a pair of RS-485 transceivers. The Mbed interfaces with external devices either through a wired USB connection or an HC-05 Bluetooth module. Figure 1(c) depicts a 120×80 mm PCB manufactured via chemical etching that accommodates the components and their associated electronics.

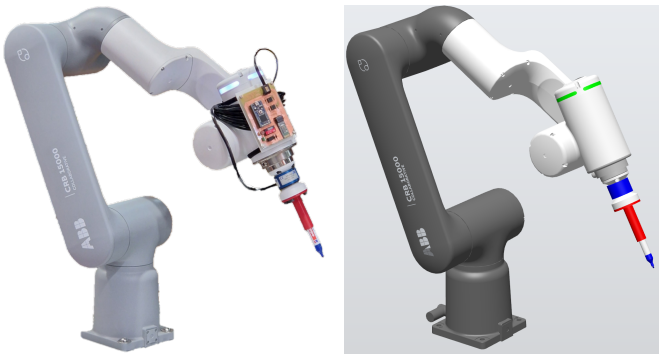


(a) Force-torque sensor (b) Pen tool (c) PCB

Figure 1: Tools mounted on the end-effector flange.

3.2. ABB "GoFa" collaborative robot

The ABB CRB 15000-5 "GoFa" collaborative industrial manipulator is the robotic platform adopted in the proposed architecture, featuring six position-controlled axes and intrinsic safety measures enabled by a torque monitoring system implemented on each joint. Although it is possible to estimate the contact forces and torques measured on the tool tip by means of said torque measuring units, better results are expected with the external JR3 force-torque sensor mounted on the tool flange. Figure 2 depicts the GoFa robot with the pen-sensor tooling attached to it, and its virtual model in the RobotStudio simulator.



(a) Real robot (b) RobotStudio model

Figure 2: ABB CRB 15000-5 "GoFa".

3.3. Grasping tools

In addition to the main drawing task, kinesthetic guidance was devised as an alternative means to test the software implementation through direct and constant user supervision. It was aimed to help in the validation of tool compensation routines and high-frequency motion commands prior to executing trajectories on a predefined path. For that purpose, a handle-like tool was obtained through resin 3D print, for improved robustness, and attached to the tool flange as shown in Figure 3.

(a) 3D-printed handle (b) Grasping-ready GoFa robot

Figure 3: Main components of a grasping robotic setup.

4. System architecture

4.1. Externally Guided Motion (EGM) and ROS 2 layers

This work expands on the architecture proposed in earlier studies (Łukawski et al., 2025a). A distributed network of software and hardware elements is arranged to take advantage from reusing existing components, to leverage available resources, and to facilitate expanding the system with new enhancements in future developments. Well-established frameworks and specialized controller capabilities are empowered to build new low- and high-level applications on top of the base hardware building blocks introduced in the previous section.

Figure 4 depicts the proposed system architecture and outlines the connections between its fundamental components.

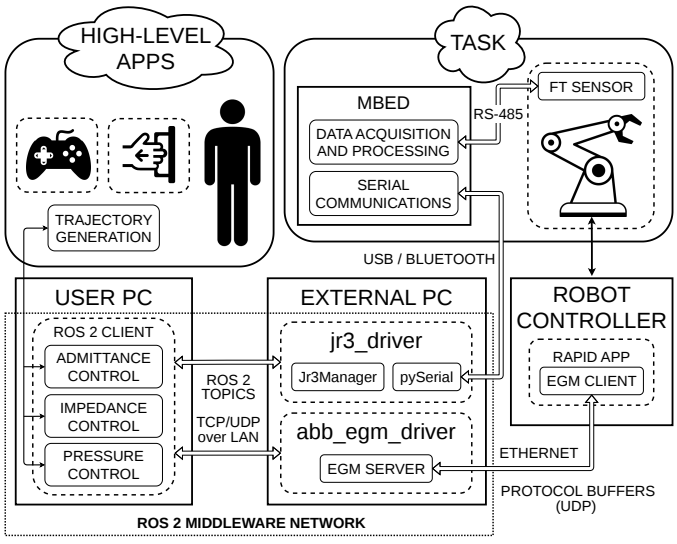


Figure 4: Proposed system architecture.

The system aims to adopt force feedback in position-controlled tasks, using the force-torque JR3 sensor mounted on the ABB GoFa's tool flange to directly measure external reaction forces being exerted by the environment on the robot. Two main tasks are devised, supported by the available tooling attached to the robot's end effector: manual grasping and drawing. While the former assumes that a human operator would perform kinesthetic guidance to manually place the tool at the desired pose, the latter can be executed either autonomously (which is in the scope of this paper) or via teleoperation. Therefore, these tasks rely on the high-frequency generation of robot motion commands, originating from two sources: either the human operator exerts an external force to manually guide the robot, or a trajectory generator produces a predefined path while applying corrections calculated using JR3 force-torque sensor's measurements.

At the core of the system lie two communications frameworks. Robot Operating System 2 (ROS 2) is a widely-adopted open-source robotics middleware that leverages TCP/UDP protocols to enable a distributed network of software components connected through communication channels named "topics", that is, ports in ROS jargon. It is a de facto standard in the robotics community and has seen progressive implantation in the industry in recent years. On the other hand, Externally Guided Motion (EGM) is an accessory feature of ABB's RobotWare controller to achieve high-frequency (250 Hz) UDP data transfer for state and motion command streams, implementing Google's protobuf library for language-agnostic message definition and transmission. Besides ABB's proprietary programming language RAPID used for initiating EGM communication at the client side (robot), Python and its ROS bindings have been chosen as the primary medium for developing the described software components.

Force-torque data acquisition occurs at the Mbed microcontroller, which receives data and clock serial streams via RS-485 transceivers from the JR3 sensor. Incoming pulses are processed and filtered, then transmitted through a serial bus to an external device. To avoid obstruction, the onboard Bluetooth module can replace USB wires for that purpose.

On the external device, the ROS 2 "jr3_driver" package instantiates a serial channel to communicate with the Mbed through the custom "Jr3Manager" class. Additionally, the "abb_egm_driver" ROS 2 package starts an EGM communication channel with the robot controller through UDP. A simple RAPID application must be running on said controller to receive and process incoming commands, and to stream robot state to the EGM server executed on the external device. If working in simulation, the virtual robot controller runs on the same device within a RobotStudio simulation instance.

Both ROS 2 processes allow to expose sensor data acquisition routines and motion control commands through standard ROS 2 interfaces and messages, thus enabling code reuse. By leveraging ROS 2 middleware capabilities, a higher-level application materialized as a ROS 2 client, interfacing with said processes through ROS 2 topics, might be executed on the same machine as those or on a different computer connected to the same local network. This capability paves the way to integrating components with specific hardware requirements, and introducing teleoperation scenarios.

Table 1: Serial protocol adopted for interfacing with the Mbed controller.

command	code	direction	payload (bytes)	description/details
acknowledge	<01>	out	1	status: 0x00 – sensor ready, 0x01 – not initialized
start	<02>	in	6	2 LSB bytes: low-pass filter cutoff frequency in 0.01 Hz (integer) 4 MSB bytes: period in μs (integer)
stop	<03>	in	0	despawn sensor acquisition and publish threads
zero offsets	<04>	in	0	all FT values are set to zero and subtracted from next measurements
set filter	<05>	in	2	low-pass filter cutoff frequency (see "start")
get state	<06>	in	0	retrieve sensor state (ready, not initialized)
get full scales	<07>	in	0	calibration data as queried from the sensor's firmware
reset	<08>	in	0	restart Mbed and acquire calibration data from sensor
read FT data	<09>	out	14	(6x) 2 LSB bytes: Fx, Fy, Fz, Mx, My, Mz (integer, signed) 2 MSB bytes: frame counter (integer, unsigned)
bootup	<10>	out	0	indicates that the sensor has started up

4.2. Serial protocol for Mbed-PC communication

The underlying data acquisition and processing (including low-pass filtering) of raw sensor data was described in a previous work (Łukawski et al., 2024). The original Controller Area Network (CAN) communication protocol between the Mbed and the TEO humanoid robot has been adapted to expose a similar set of instructions over a serial channel (USB or Bluetooth) to interface with a wider range of external devices. Table 1 lists the supported commands with their associated code, directionality, data payload and description. The associated C++ implementation of Mbed's firmware has been uploaded to a public GitHub repository.[1]

A separate protocol was developed to describe messages flowing between the Mbed and the external PC through a serial channel. Each message is delimited by the "<" and ">" start and end characters, respectively. A two-character wide numeric command code is placed between those delimiters, followed by a binary byte array describing the message payload, if supported. Command codes are string-encoded to facilitate manual interpretation and handling. Messages are stored in receive and transmission buffers and can be read and sent in batches for improved performance.

To ensure robustness, all incoming commands (marked as "in" in the previous table) generate an acknowledge message right upon reception, to be transmitted to the external device and processed by the "Jr3Manager" class.

4.3. Jr3Manager Python library

On the external device side, a client Python library has been introduced to communicate with the Mbed via a serial channel. The command set was translated into Python code for ease of development of new high-level applications, such as the "jr3_driver" ROS 2 package introduced in this work. Strict compatibility has to be maintained with the protocol implemented on the Mbed side.

Figure 5 depicts the Unified Modeling Language (UML) class diagram of the "Jr3Manager" class, showcasing its Application Programming Interface (API). Python-specific features have been leveraged, such as tuple data structures.

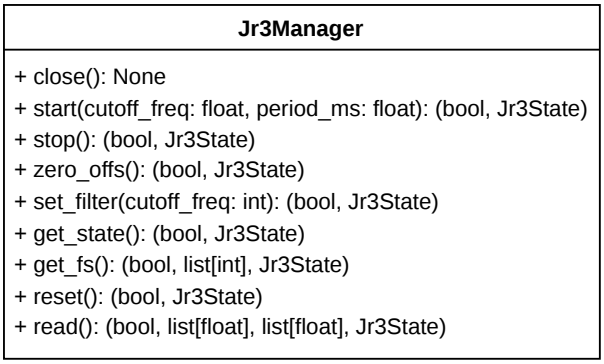


Figure 5: UML class diagram of the Jr3Manager Python class. Jr3State is an enumeration exposing the "ready" or "not initialized" states.

Aside from the "close" method, devised to gracefully join the internal read thread and close the serial channel, most methods produce a RPC-like (Remote Procedure Call) request that awaits an acknowledged response (<01> command code) from the Mbed with a predefined timeout. If successful, among other data if applicable, each method signals a Boolean "true" as the first member of the returned tuple, along with the Mbed controller state ("ready" or not "initialized") encoded in the "Jr3State" enumeration.

When "start" is invoked, a background thread is started to process a constant stream of FT data arriving from the Mbed. The "read" method queries the last FT value stored internally.

4.4. ROS 2 packages

The new "jr3_driver" ROS 2 Python package accommodates an instance of "Jr3Manager" in order to expose its interface over the ROS 2 network. For compliance with client applications, the standard "geometry_msgs/Wrench" message type was adopted. In addition to the main publisher node, the package also hosts high-level applications that rely on it. Besides, it contains a trajectory generation module derived from the original C++ implementation of Orocos Kinematics and Dynamics Library (KDL), which lacks its corresponding Python bindings at the time of writing. The source code is available on GitHub.[2]

[1] https://github.com/roboticslab-uc3m/jr3-mbed-firmware-uart
[2] https://github.com/roboticslab-uc3m/jr3_driver

Another ROS 2 package, "abb_egm_driver", was introduced to expose EGM commands over the ROS 2 network. Its main node depends on the "ABBRobotEGM" Python library contributed by F. Lobert. All EGM capabilities are leveraged by this dependency: position state streaming (from the robot to external devices), position guidance (for sending motion commands from external devices to the robot), and path correction (for tweaking a coordinate based on the measurements of an external device such as a force-torque sensor).[3]

Similarly, standard ROS 2 message types have been chosen such as "geometry_msgs/Pose" to denote Cartesian position and orientation. The source code has been uploaded to GitHub.[4]

5. Applications and experiments

5.1. Validation in a simulated environment

Prior to conducting experiments on a real GoFa robot, the "Jr3Manager" library and the Mbed firmware were tested on a virtual environment using the RobotStudio simulator. A simplified setup was prepared consisting on the electronics and a self-standing JR3 sensor, whereas the chosen robot was an ABB IRB 1200 model (reach: 0.9 m, payload: 5 kg). The scenario consisted of a predefined rectangular path on the horizontal plane which the robot needed to follow cyclically with an attached training tool. During the execution of the trajectory, the user exerted a force by pressing the mounting flange of the force-torque sensor, thus emulating an obstacle that a real robot would sense thanks to this sensor being mounted at its flange. The application applies a correction in the vertical direction upwards, proportional to the force exerted by the user.

Figure 6 depicts the setup. The middle plot represents live Z-axis force measurements obtained from the sensor, which are mapped to the height (along the positive Z axis) the robot is commanded to lift while the user presses the sensor.

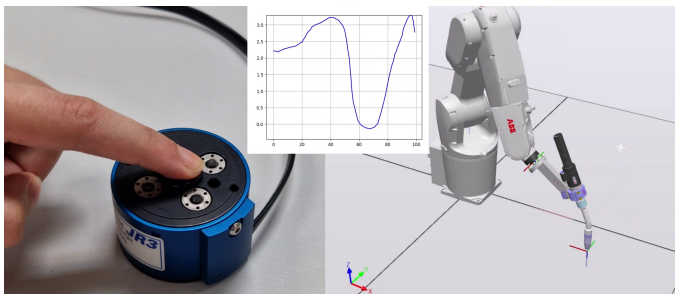


Figure 6: Path correction with force feedback on a simulated ABB robot.

5.2. Admittance and impedance control

Once mounted on the robot's flange, the sensor and accompanying 3D-printed adapters, along with the tool itself, must be treated as a fixed link with a non-negligible mass that need to be accounted for by the system. Otherwise, said mass would be added to the external forces and moments detected by the sensor during the interaction with the environment.

Algorithm 1 describes the compensation routine that removes the contribution of the tool from the sensor measurements. Given the gravity constant g and tool mass m, the wrench ${}^{0}\mathcal{W}$ expresses its weight in the inertial frame 0 (robot base). The vector ${}^{S}\boldsymbol{p}_{CoM}$ defines the center of masses of the tool in the sensor frame S. This routine uses the current joint configuration $\boldsymbol{\theta}$ to obtain the pose of the tool center point (TCP) ${}^{0}H_{TCP}$ via forward kinematics, which is pre-multiplied by the transformation between the sensor and the TCP frames ${}^{S}H_{TCP}$ to find the tool wrench expressed in the sensor frame, ${}^{S}\mathcal{W}$, without affecting its reference point (only rotations are applied). Its 3-axis moment vector ${}^{S}\mathcal{W}\|\boldsymbol{m}$ is multiplied by ${}^{S}\boldsymbol{p}_{CoM}$ to express the moment of the tool's weight at its center of masses. Finally, said weight ${}^{S}\mathcal{W}$ and the initial sensor measurements ${}^{S}\mathcal{F}\big|_{t=0}$ are subtracted from the current sensor measurements ${}^{S}\mathcal{F}$ to obtain the actual external forces and moments exerted at the origin of the sensor frame.

Algorithm 1: Tool compensation routine.

Data: g, m, ${}^{S}H_{TCP}$, ${}^{S}\boldsymbol{p}_{CoM}$
${}^{0}\mathcal{W} \leftarrow [0, 0, -g \cdot m]$
Function `compensate_tool`$(\boldsymbol{\theta}, {}^{S}\mathcal{F}, {}^{S}\mathcal{F}\big|_{t=0})$**:**
  ${}^{0}H_{TCP} \leftarrow forward\_kinematics(\boldsymbol{\theta})$
  ${}^{S}\mathcal{W} \leftarrow {}^{S}R_{TCP} \cdot ({}^{0}R_{TCP})^{-1} \cdot {}^{0}\mathcal{W}$
  ${}^{S}\mathcal{W}\|\boldsymbol{m} \leftarrow {}^{S}\mathcal{W}\|\boldsymbol{m} + {}^{S}\mathcal{W}\|\boldsymbol{f} \cdot {}^{S}\boldsymbol{p}_{CoM}$
  ${}^{S}\mathcal{F} \leftarrow {}^{S}\mathcal{F} - {}^{S}\mathcal{W} - {}^{S}\mathcal{F}\big|_{t=0}$
  return ${}^{S}\mathcal{F}$

Since the center of masses of the assembly is a rough estimation, non-zero force-torque measurements may occur after applying the compensation, even though no external contacts are registered. To account for these spurious readings, a small deadband is introduced and compared against the norm of the force and moment vectors separately, assuming any measurement is to be considered null if falling within said deadband.

The previous features were implemented in an admittance control application on the handle tool setup to ensure proper communication between all involved components, and to validate the tool compensation routine. It was deemed cautious to test the setup in a user-controlled kinesthetic guidance scenario prior to letting a trajectory generation module and the force compensation take over the motion of the robot while being uncertain about the outcome of the intermediate calculations.

The admittance controller builds a displacement vector from the compensated force-torque measurements in the tool frame, acquired through the "jr3_driver" node, and sends a position command via EGM and the "abb_egm_driver" node to the robot. A factor can be applied to the linear and angular motion exerted by the user on the handle, before translating it to commanded robot motions. As a variation of this, an impedance control application was derived to implement a spring-like behavior with optional damping.

Both applications have been tested on the GoFa robot. The estimated mass and center of masses have been verified, and the software architecture has been validated against a real robot.

[3] `https://github.com/FLo-ABB/ABB-EGM-Python`
[4] `https://github.com/roboticslab-uc3m/abb_egm_driver`

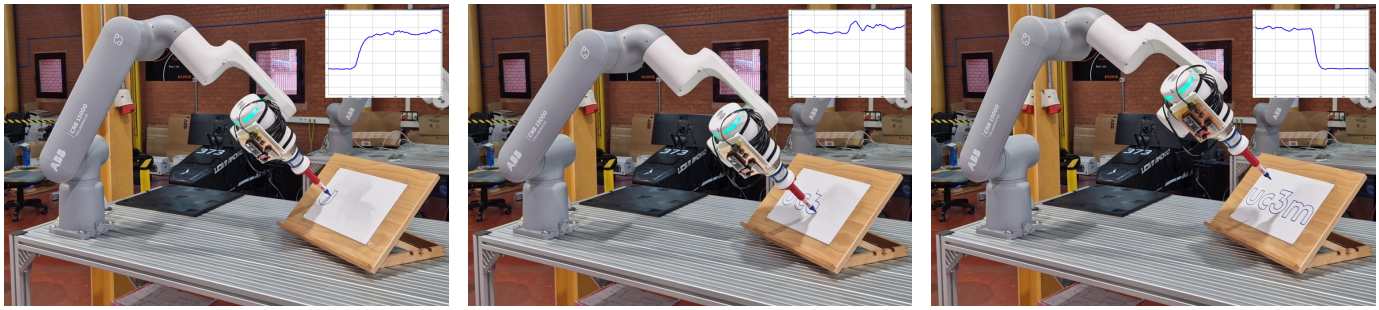


Figure 7: Drawing task with path correction using external force-torque feedback. The normal force component is depicted using the "rqt plot" tool.

5.3. Drawing task with controlled pressure

A third high-level application was developed to fully test the behavior of the proposed force-feedback architecture on a predefined complex path. The targets of the trajectory were programmed and stored in the robot controller as RAPID code, then connected through precise `EGMMoveL` and `EGMMoveC` linear and circular motion instructions, respectively. The RAPID program runs an EGM client that expects path corrections to the Z component to be supplied by the application. Sensor measurements, provided by the "jr3_driver" node and compensated thereupon, are used to calculate said component and send it to the robot controller through the "abb_egm_driver" node. The path correction ensures that the tool exerts a configurable and fixed pressure on the paper surface.

Figure 7 shows a sequence of the experiment during which the attached drawing tool was used to reproduce a non-continuous regular sketch with straight lines and circular arcs. The trajectory involves lifting the pen and moving it to a different position in order to complete the drawing. The application was validated with multiple trajectory speeds, up to a velocity of the tip of 500 mm/s.

6. Conclusions

This work presents a controller architecture for EGM-compatible ABB robots, specifically the CRB 15000-5 "GoFa", integrating a JR3 six-axis force-torque sensor managed by an Mbed NXP LPC1706 controller, and orchestrated by the ROS 2 middleware through newly contributed Python-based nodes.

The proposed setup and its hardware and software components have been validated through two high-level tasks: manual (kinesthetic) guidance and execution of predefined trajectories with path compensation. Experiments have been conducted with real sensor data both on a virtual model in RobotStudio and the real robot.

Future work is expected to extend the capabilities of the system with the teleoperation architecture introduced in previous works (Łukawski et al., 2025b). Haptic feedback can be taken advantage from to provide the user with improved sensing of the environment the robot is interacting with. For instance, a 6-axis haptic device could be integrated to teleoperate the robot and to draw on a surface using the pen tool. Sensor measurements can be used to generate a feedback on the device so that the operator is able to perceive the reaction forces of the environment on the tool, and apply the right pressure over the drawing surface during the task.

Acknowledgments

This research has been financed by "FotoArt5.0-CM, Laboratorios inteligentes para la ciencia del futuro" (TEC-2024/TEC-308) and "iRoboCity2030-CM, Robótica Inteligente para Ciudades Sostenibles" (TEC-2024/TEC-62), both funded by "Programas de Actividades I+D en Tecnologías de la Comunidad de Madrid"; "ROBOASSET: Intelligent robotic systems for assessment and rehabilitation in upper limb therapies" (PID2020-113508RB-I00, financed by AEI/10.13039/501100011033); "iREHAB: AI-powered Robotic Personalized Rehabilitation" (DTS22/00105, financed by Instituto de Salud Carlos III (ISCIII)); "ASEPEYO-UC3M: FASE IV" serious game-based rehabilitation program; and EU structural funds.

References

Bagge Carlson, F., Haage, M., 2017. YuMi low-level motion guidance using the Julia programming language and Externally Guided Motion Research Interface. Technical Reports TFRT-7651. URL: `https://lup.lub.lu.se/record/eebe1d1d-474e-4cd2-b815-12d110c091c3`.

Garcia-Haro, J. M., Oña, E. D., Hernandez-Vicen, J., Martinez, S., Balaguer, C., 2020. Service robots in catering applications: A review and future challenges. Electronics 10 (1), 47.
DOI: `10.3390/electronics10010047`

Haddadin, S., De Luca, A., Albu-Schäffer, A., 2017. Robot collisions: A survey on detection, isolation, and identification. IEEE Transactions on Robotics 33 (6), 1292–1312.
DOI: `10.1109/TRO.2017.2723903`

Hernandez-Vicen, J., Łukawski, B., Martinez, S., Tsagarakis, N., Balaguer, C., 2024. Hybrid sensor fusion mixed with reinforcement learning in autonomous dual-arm lifting tasks performed by humanoid robots. Robotics 13 (8), 121.
DOI: `10.3390/robotics13080121`

Łukawski, B., Oña, E. D., Jardón, A., Victores, J. G., Balaguer, C., 2025a. Development of educational applications with ABB GoFa collaborative robot using Externally Guided Motion. In: Int. Conf. on Control, Automation and Diagnosis (ICCAD). IEEE.
DOI: `10.1109/ICCAD64771.2025.11099395`

Łukawski, B., Oña, E. D., Victores, J. G., Balaguer, C., Jardón, A., 2026. Assessing student satisfaction and system usability in industrial robotics education. In: II Simposio Conjunto de los Grupos Temáticos de CEA de Ingeniería de Control, de Modelado, Simulación y Optimización, y de Educación en Automática. No. 2. CEA.
DOI: `10.64117/simposioscea.v2i1.150`

Łukawski, B., Rebollo, M., Gilabert, Á., Victores, J. G., Balaguer, C., Jardón, A., 2025b. YARP Cartesian controller layers over ROS 2 for teleoperation and web applications. In: Jornadas de Automática. No. 46. CEA.
DOI: `10.17979/ja-cea.2025.46.12252`

Łukawski, B., Rodríguez-Sanz, A., Victores, J. G., Balaguer, C., 2024. An open-source implementation of a force-torque sensor data acquisition device for the humanoid robot TEO. In: Simposio de Robótica, Bioingeniería y Visión por Computador. CEA, pp. 79–84.

Monje, C. A., Martinez, S., Pierro, P., Balaguer, C., 2018. Whole-body balance control of a humanoid robot in real time based on ZMP stability regions approach. Cybernetics and Systems 49 (7–8), 521–538.
DOI: `10.1080/01969722.2018.1552858`

Obal, P., Gierlak, P., 2021. EGM toolbox – interface for controlling ABB robots in Simulink. Sensors 21 (22).
DOI: `10.3390/s21227463`

Pérez Martínez, C., Pierro, P., Martinez, S., Pabon, L., Arbulú, M., Balaguer, C., 2010. RH-2: an upgraded full-size humanoid platform. In: Mobile Robotics: Solutions and Challenges. World Scientific, pp. 471–478.
DOI: `10.1142/9789814291279_0058`