IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 5, 2017, accepted January 18, 2018, date of publication January 24, 2018, date of current version March 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2797803

An FPGA-Based Open Architecture Industrial Robot Controller

MIGUEL-ANGEL MARTíNEZ-PRADO, JUVENAL RODRíGUEZ-RESÉNDIZ, (Senior Member, IEEE), ROBERTO-AUGUSTO GÓMEZ-LOENZO, GILBERTO HERRERA-RUIZ, AND LUIS-ALFONSO FRANCO-GASCA

Facultad de Ingeniería, Autonomous University of Querétaro, Santiago de Querétaro 76010, Mexico

Corresponding author: Juvenal Rodríguez-Reséndiz (juvenal@uaq.edu.mx)

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología and in part by PRODEP.

ABSTRACT Most robot controllers are proprietary and, therefore, they have a closed architecture, and it is typical for robot vendors not to provide support for old control units; hence, when a deprecated robot requires an update or even a simple preventive maintenance, the manufacturer tends to recommend disposing such a unit and to acquire a new one. In this paper, the development of a field programmable gate array-based motion control system is described as part of an open-architecture and vendor-independent control system tested on a Fanuc S420F. As a way to measure the performance of the developed system, an experimental test is carried out to achieve a path tracking RMS error lower than 20 mm at a Cartesian velocity of 500 mm/s. Results demonstrate that the proposed system achieves the same feed rates as the original controller with the significant advantage of being modular, flexible, scalable, and reconfigurable.

INDEX TERMS Control-retrofit, FPGA, industrial-robot, motion-controller, open-architecture, servo-control.

I. INTRODUCTION

The mechanical part of industrial robots has not evolved at the same rate as its electronics. Nowadays, robot control systems are faster, smaller, and even they incorporate sensory information such as force and vision. These technological advances have permitted the number of applications of such machinery to become widespread [1]; however, the closed architecture of their control systems involves a continuous dependency of the end-user on the (particular) robot vendor [2].

The above problem surfaces when maintenance, update, or the incorporation of new functionality to a working—and usually deprecated—unit is required. This is because most of the robot manufacturers do not provide support to deprecated control units and their controllers have limited scalability. A modular, open-architecture [3] and vendor-independent control system is required to solve this problem.

Nowadays, there is still not a standard robot control architecture; there are some formal proposals for standardizing robot control architectures, but in practice there is no de facto standard and each control system has a unique architecture, as is the case with most home-made or customized low-performance CNC machinery systems and 3D printers [4]; however, a robot control system as the one mentioned above implies basically the development of three main components: a hardware platform, an operating system (OS)-based module, and a software application as Hong *et al.* describe in [5].

The hardware platform comprises as a minimum the following instances: robot (mechanical systems) and actuators, power amplifiers, data acquisition boards, servo control components, and a master device for coordination of tasks as it is addressed in [6].

Computer controlled hardware requires a way to communicate with the computer. OS modules are conceived as hardware drivers. It is mandatory that such drivers have a real-time (RT) behavior for specific applications such as robot trajectory generation since it is planned on the fly [7]. Those drivers are responsible for performing data acquisition, computing interpolation and solving inverse kinematics; on top of these drivers, the application software serves as a human-machine interface that enables the user to coordinate with the robot controller by entering commands using both a friendly interface and a specific-purpose programming language.

Servo control is the most critical part in a robot control in the sense that its failure to maintain a closed-loop behavior may lead to a halt in the production process, the destruction

of the robot itself or, in the worst-case scenario, the loss of human lives. Therefore, most researchers have implemented robot control systems with mainly three kinds of servo control solutions, namely: standard/industrial PC with data acquisition boards, digital signal controllers (DSC), field programmable gate arrays (FPGA), or any combination of them such as the programmable system on a chip (SoC) technology.

II. RELATED WORK

Since the decade of the 90s, researchers have been developing customized robot controllers to avoid the problems mentioned above. For instance, Ferretti *et al.* [8] implemented a methodology [9] to extend the functionality of a Comau C3G-9000 robot controller. They incorporated sensor boards to the system which are connected to the main control units through a common bus. Both units, robot controller and servo control, include several microprocessors and digital signal processors (DSP). The objective of the authors was to perform rubbing tasks; however, they do not report any results of the developed robot controller performance. Furthermore, the cost of a multi-processor based solution is considerably higher compared with the reconfigurable logic technology. A simplified solution for the same robot controller is found in [10]. This latter research work deals with the implementation of a robot controller within the Linux RTAI OS [11]; however, as in the case above, the authors do not include information about the system performance.

Another similar hardware architecture is described in [12] to control an ABB industrial robot for spot and arc welding applications. Here, the authors propose a complete architecture for efficiently programming robotic systems. Nevertheless, given that the hardware architecture is also based on a hybrid multi-processor and DSP system, the proposed architecture is focused only on software layers, which does not allow the controller to exploit reconfigurable logic hardware techniques. Besides, the authors do not prove in a quantitative way that the suggested architecture streamlines the programming process.

In [5] Hong *et al.* implemented a PC-based robot controller following the OSACA approach [13]; the hardware module of the robot control system comprises a standard PC running Windows NT and a *Delta Tau PMAC* [14] motion controller card, and the other two components are hardware drivers and the application software. In contrast with previous research works, Hong *et al.* do include information related to the system performance; nevertheless, the software interface only supports off-line programming because the operating system has no RT capabilities and, therefore, it is necessary to compute the coordinate transformations for the whole trajectory before its execution.

In [15], a group of undergraduate students carried out a hardware retrofit of a *Unimate* PUMA 560 robot by replacing the VAL II control system with a standard computer and a data acquisition board. The software architecture was based on MATLAB/Simulink and ran under Windows 2000 and WinCon for RT performance. However, the WinCon platform does not guarantee RT task scheduling, and hence, a more demanding application cannot be successfully performed. Another similar work is described in [16]. The authors replaced the VAL II of a PUMA robot with a PC based controller; however, as in the case above, the developed system does not involve an RTOS and only can be used with robots actuated by DC brushed motors.

In [17] a visual servoing system is implemented on an FPGA to control a four DoF robot. Same case as above, the motors of this robot are DC brushed motors and they do not require any kind of external commutation; therefore, this solution can not be used for industrial robots such as the Fanuc S420.

In this sense, most authors affirm that an industrial robot controller must have an RT behavior. In [7], they developed a robot control system (ROACS) for a PUMA 560 to cut fish slices, and claim that within an uncertain environment, the robot control system must be able to change its path according to feedback sensory information. The ROACS is software-based using the QNX real-time operating system (RTOS).

In [18] Visioli and Legnani experimented with a SCARA industrial robot. They implemented several control schemes with the aid of a standard PC and the QNX RTOS; nevertheless, their controller only concerns to position control, and they do not include any other feature of a whole robot controller.

Similarly, Chang *et al.* [19], using a *Texas Instruments TMS320C3X* from the DSP family, implemented a position control loop with a digital filter in the input to avoid vibration due to slewing motion for a linear motor. Although the DSP platform presents several processing advantages over a standard PC, the *TMS320C3X* device family only has two encoder quadrature interfaces; therefore, an industrial robot with six degrees-of-freedom (DoF) requires at least three devices which increases the cost of the solution and its hardware complexity.

An interesting control architecture is described by Shao and Sun [20], involving a mixed DSP-FPGA digital platform for the motion control of a two DoF Yamaha robot; they report that their system exhibits a better performance than the original control system. Similar work can be found in [21] and [22]. In both research works, the complexity of interpolation and kinematics is delegated to the DSP; a standard PC with an RTOS replaces the DSP which leads to having a system with a higher degree of openness [23].

Modern motion control implementations are based on FPGAs because of the advantages they have over DSPs and micro-controllers [24]. In [25] an FPGA-based motion controller is developed to control the position of an XY table actuated by permanent-magnet synchronous motors (PMSM); however, this research work only deals with two axes, and the amount of logic elements it utilized makes it difficult to implement such motion controller in most low-density FPGA families. Furthermore, the usage of the Nios

TABLE 1. Former robotic research projects.

Author	Robot/controller	Technology	RT support
Ferreti *et al.*	Comau C3G-9000	DSP	No
Macchelli and Melchiorri	Comau SMART 3-S	PC	Yes
Nilsson and Johansson	ABB IRB-2000	DSP	No
Hong *et al.*	SCARA	PC and PMAC	No
Becerra *et al.*	PUMA 560	PC	No
Farooq and Wang	PUMA 560	PC	No
Visioli and Legnani	ICOMATIC03 SCARA	PC	Yes
Chang *et al.*	Adept-linear-robot	DSP	No
Shao and Sun	SCARA	DSP/FPGA	No
Liandong *et al.*	Movemaster-EX robot	PC	No
Kung*et al.*	2-axis linear robot	FPGA	No

microprocessor limits its implementation to only a few device families.

Most of the related research work deals with small-size robots such as the SCARA, PUMA, and others mostly actuated by direct current (DC) brushed motors as it is summarized in Table 1. In contrast, the brushless motors of bigger robots require external commutation which leads to a major complexity in the motion control system.

Up to date, there is no evidence of the development of a functional control system suitable for robots such as the Fanuc S420F. The aim of this paper is to contribute to this gap with an FPGA-based motion controller for industrial robots actuated by PMSM and servo amplifiers with external commutation.

III. METHODOLOGY

The industrial robot used in this research work was disposed from a factory and acquired by Universidad Autónoma de Querétaro as a second-hand industrial system, and no control unit was included with the purchase; therefore, the aim of this project was to design a new control system with modern technology in order to reassemble an operational robotic cell for arc welding processes. The control system had to accomplish three primary requirements: low cost, open architecture, and a path tracking error lower than 2% at a Cartesian velocity of 500 mm/s.

The proposed robot control architecture comprises a standard PC and an FPGA-based interface which involves a lower cost. Additional cost saving is achieved by using the open-source Linux OS in addition with the real-time application interface (RTAI) patch for RT scheduling of tasks.

The FPGA is responsible for decoding feedback position data, computing the control output and commutation signals, and transmitting such data to the actuators, whereas the PC interpolates movements, computes kinematics solutions, and transmits the position reference for each joint while it receives its current position. The entire system is illustrated in Fig. 1.

The modules within the system have been arranged according to their RT requirements as shown in Fig. 2, being the most critical tasks implemented in hardware, whereas non-critical tasks such as user-interface and robot animation remain as simple threads in the user-space of the Linux kernel.

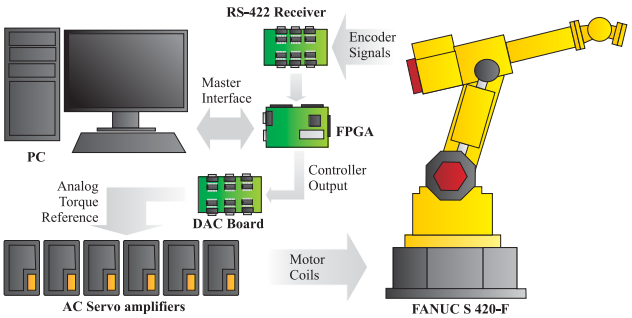


FIGURE 1. Robot controller hardware setup.

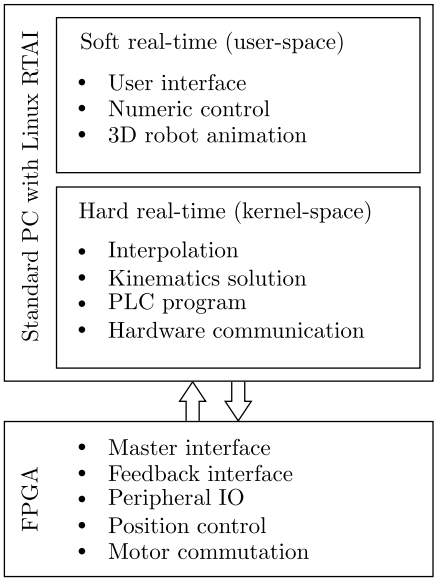
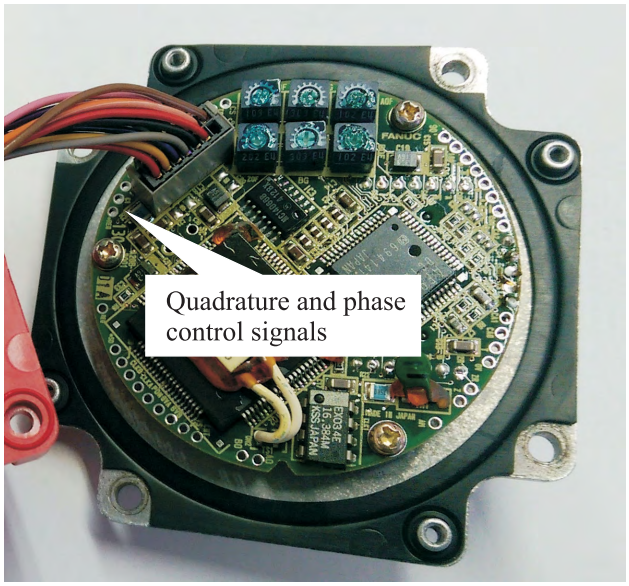


FIGURE 2. Robot control system layers.

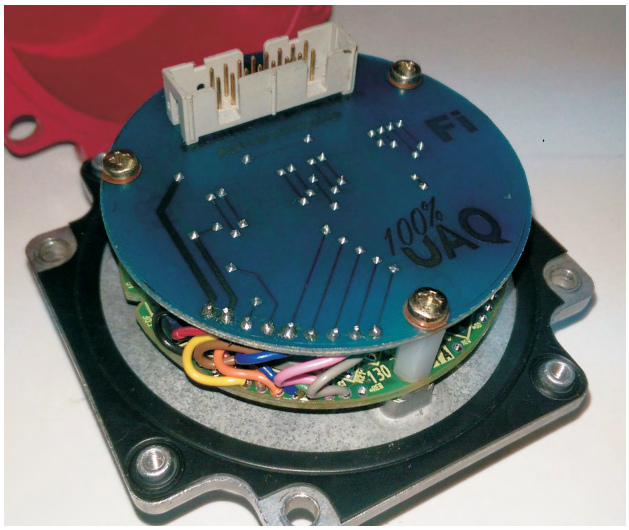
A. INSTRUMENTATION

A reverse engineering process was carried out in order to identify the signals in the feedback encoder for each of the robot motors. Quadrature and phase control signals are available in the PCB of the feedback encoder as illustrated in Fig. 3a.

The quadrature signals are decoded for position feedback whereas phase control signals are used for commutation. With the aim of preserving the integrity of the signals on



(a)



(b)

FIGURE 3. Instrumentation of the feedback device: (a) original device and (b) modified version.

the transmission line, an additional PCB with a couple of *DS34C87TM* RS-422 line drivers has been attached to the feedback device as depicted in the Fig. 3b. On the other hand, a PCB with *DS34C86* RS-422 line receivers is connected to the FPGA board.

The Fanuc S-420F is actuated by AC permanent magnet synchronous motors (PMSM); therefore, six *Advanced Motion Controls* servo amplifiers with external commutation [26] are utilized as power stage.

These servo drivers require two analog current references for two of the motor coils while the third one is computed internally. Current references are given as voltage, i.e., a set-point of $+10\,\text{V}$ represents a current of $+30\,\text{A}$ at a given motor coil. Two instances of the *Analog Devices* digital to analog converter DAC5668 are used to generate such current

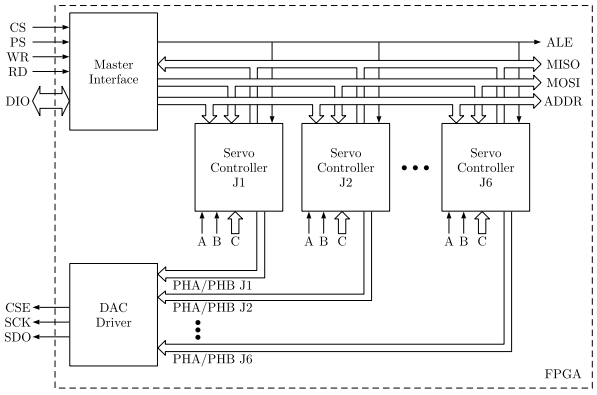


FIGURE 4. Architecture of the motion controller circuit.

references; however the DAC output ranges from 0 V to 5 V; therefore, another PCB includes a set of operational amplifiers to rearrange the output from $-10\,\text{V}$ to $+10\,\text{V}$.

B. HARDWARE DESCRIPTION

The motion controller circuit includes mainly a master interface circuit, several instances of a servo controller module and a digital to analog converter driver as illustrated in Fig. 4.

The *Master Interface* module serves as an interface between the PC and the motion controller. On the PC side, it has a parallel bus with an eight-bit width, and it is designed to be used with the LPT port due mainly to the following reasons:

- Synchronization of the PC and the FPGA. By using this port, it is possible to assure that the transfer of a word is always done at the required time instant.
- The parallel port is more reliable under high electrical noise conditions as in this case.
- A baud rate of 1.2Mbps is achieved with the enhanced parallel port (EPP), which is fast enough to send and receive the reference and position data of the six robot joints.
- And finally, because of its simplicity and versatility as it is documented in [27].

There are other and more common ways to link an external system to a PC, but this is done as a proof of concept showing the modularity of our proposed architecture. Moreover, in [28], the EPP, UART, and Ethernet ports are evaluated, obtaining the fastest baud rate with the parallel port.

Additionally, to the *DIO* data bus, there are four control signals, namely, *CS* (chip select), *PS* (port select), *RD* (read), and *WR* (write) as illustrated in Fig. 5.

It is necessary to enable *CS* to start a data transaction. The signal *PS* indicates the type of transaction; there are two types of transactions: data and command transactions—the data transactions are referred to the values of the registers, while command transactions determine whether it is a read or write process and the destination address. The PC or master device transfers a byte each time the *RD* or *WR* are asserted.

Given that the *Master Interface* module has a shift register internally for reading and writing operations, up to four bytes may be transferred in one single data transaction.

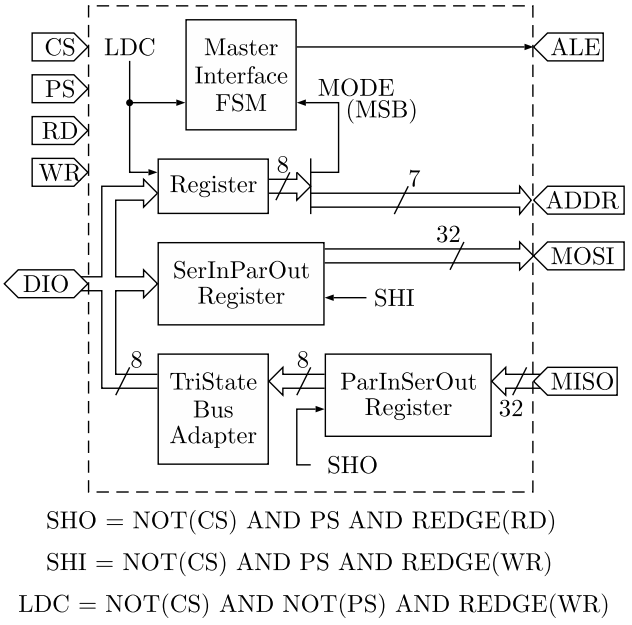


FIGURE 5. Block diagram of the ***Master Interface*** module.

On the other hand, there are three buses and one control signal. *MISO* and *MOSI* have 32 bits of width, and they are used for data interchange. The bus *ADDR* is used to select the *Servo Controller* instance and the register. The signal *ALE* (address latch enable) acts as a trigger for the data interchange, i.e., the register with the specified address stores data when this signal is asserted.

The above process is possible since each *Servo Controller* instance has a unique address. The three most significant bits of the *ADDR* bus indicate the number of *Servo Controller* instance while the last four indicate the parameter to be accessed.

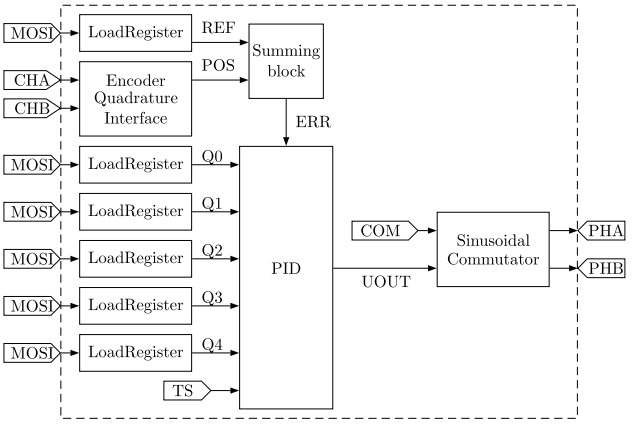


FIGURE 6. Block diagram of the ***Servo Controller*** module.

The *Servo Controller* module is composed of an encoder quadrature interface, a summing block, a proportional-integral-derivative (PID) controller, a sinusoidal commutator, and several registers used to store reference and controller parameters as illustrated in Fig. 6.

The *Encoder Quadrature Interface* decodes the mechanical position of each joint from A and B quadrature signals. The output of this module has a width of 32 bits. The position signal (POS) is subtracted to the set point signal (REF), and it is adjusted to 16 bits of width. The above to avoid saturation in forward computations. The resulting signal represents the position error, and it is labeled as ERR in Fig. 6.

The *PID Controller* is implemented as a fourth-order infinite-impulse-response (IIR) filter as that one described in [29] but without a derivative low-pass filter. The implemented formula is

$$u(k) = K_p\Big\{e(k) + \frac{T_s}{T_i}\sum_{i=1}^{k} e(i) + \frac{T_d}{T_s}[e(k)+3e(k-1)-3e(k-2)-e(k-2)]\Big\}, \quad (1)$$

where K_p, T_i, T_d, and T_s are the proportional gain, integral time, derivative time, and sampling time, respectively. To avoid the storage of the error samples from time $t = 0s$ until $t = kT_s s$ and with the help of the relationship $\Delta u(k) = u(k) - u(k-1)$, a suitable function may be written as:

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + q_3 e(k-3) + q_4 e(k-4) + u(k-1), \quad (2)$$

where

$$q_0 = K_p\Big(1 + \frac{T_s}{T_i} + \frac{T_d}{6T_s}\Big),$$

$$q_1 = -K_p\Big(1 - \frac{T_d}{3T_s}\Big),$$

$$q_2 = -K_p\frac{T_d}{T_s},$$

$$q_3 = K_p\frac{T_d}{3T_s},$$

$$q_4 = K_p\frac{T_d}{6T_s}.$$

These coefficients are given in a fixed point 16.16 format whereas the error and output signal has 16 bits of width. The *PID Controller* computes the control signal UOUT with the aid of a multiplier-accumulator (MAC) unit as illustrated in Fig. 7. The control frequency is fixed at 1 KHz.

Most of the dynamic parameters of the robot are unknown; hence, it is not possible to tune-up the controller with an analytical approach. Methods such as the Ziegler-Nichols require taking the system to the stability limit, which is not recommended for a robot since it could be seriously damaged. Therefore, an empirical method is used to tune-up the PID controller for each joint. First, the proportional gain is set in order to reach the set-point within a short time; then, the derivative time is increased to provide a well damped response; and finally, the integral time is decreased to achieve a null or very close to zero error at steady state. Table 2 enlists the obtained values for each joint.

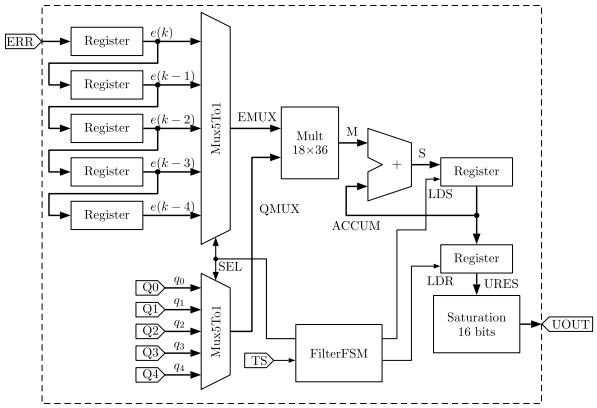


FIGURE 7. Block diagram of the PID controller.

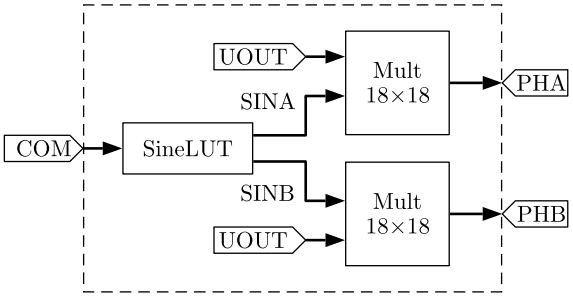


FIGURE 8. Block diagram of a *Sinusoidal Commutator*.

TABLE 2. PID parameters for each robot joint.

Joints	K_p	T_i	T_d
J1	8.0	0.15	0.12
J2	12.0	0.35	0.15
J3	16.0	0.17	0.22
J4	4.5	0.065	0.12
J5	6.0	0.065	0.12
J6	4.0	0.065	0.12

The *Sinusoidal Commutator* module reads the control output (UOUT) and the phase control signals (COM) to compute the phase current for each motor coil. Moreover, the phase currents have a sine waveform, and their amplitude depends on the value of the control signal. The frequency of these signals, commonly called *commutation frequency*, varies according to the motor velocity.

The computation of the phase currents requires the knowledge of the position of the poles with respect to the stator fluxes. The feedback device has an absolute encoder for this purpose, and its signal is labeled as COM in Fig. 8. The phase control signal has four bits of width, and it is used along with a look-up table to generate sixteen points of a sine wave.

The decoded sine value has eighteen bits, and it is given in a 2.16 format as listed in Table 3. The phase currents (PHA and PHB) are computed by multiplying the sine value times the filter output, which is internally extended also to eighteen bits. The second phase current PHB is shifted 112.5 electrical degrees. Both signals, PHA and PHB, are saturated and trimmed to sixteen bits.

TABLE 3. Sine look-up table for commutation signals.

COM	α (deg)	$\sin(\alpha + 90)$	$\sin(\alpha - 22.5)$
0000	157.5	11.0001001101111110	00.1011010100000100
0001	180.0	11.0000000000000001	00.0110000111110111
0011	202.5	11.0001001101111110	00.0000000000000000
0010	225.0	11.0100101011111100	11.1001111000001001
0110	247.5	11.1001111000001001	11.0100101011111100
0111	270.0	00.0000000000000000	11.0001001101111110
0101	292.5	00.0110000111110111	11.0000000000000001
0100	315.0	00.1011010100000100	11.0001001101111110
1100	337.5	00.1110110010000010	11.0100101011111100
1101	0.0	00.1111111111111111	11.1001111000001001
1111	22.5	00.1110110010000010	00.0000000000000000
1110	45.0	00.1011010100000100	00.0110000111110111
1010	67.5	00.0110000111110111	00.1011010100000100
1011	90.0	00.0000000000000000	00.1110110010000010
1001	112.5	11.1001111000001001	00.1111111111111111
1000	135.0	11.0100101011111100	00.1110110010000010

Finally, the current references PHA and PHB are sent to the DAC through a serial peripheral interface (SPI) with a serial clock frequency of 25 MHz. This module, labeled as *DAC Driver*, is at the end of the data path depicted in Fig. 4.

C. INTERPOLATION AND KINEMATICS SOLUTION

A robot control system needs a way to describe the instructions provided to the robot. Of course, the instructions given to the robot are directly related to the actions the robot can perform, so describing a robot programming language is equivalent to describing the variables that influence the operation of the robot to be controlled, positions, speeds and accelerations of the described paths.

There are different types of paths that a robot can perform according to the purpose being sought; in general terms, there are two types of displacements: (a) those intended to move the end-effector from one point to another without performing useful work but merely to bring it to a point of interest, and (b) motions destined to do some useful work such as welding or painting a surface. The first type of motion can be performed in a coordinated way in the Cartesian coordinate system (usually as linear interpolation), or in a coordinated way in the space of the robot joints; the second type of movement is always coordinated in Cartesian coordinates following a well-defined trajectory that can be linear, circular, some particular type of trajectory (e.g., elliptical or parabolic) or interpolated (e.g., Bézier curves or splines); depending on the type of trajectory, it is possible to re-parameterize the curve to obtain a concordance between the velocities and accelerations specified by a previously chosen velocity profile (triangular, trapezoidal, trigonometric, etc.). Note that the paths that describe the movement of the robot occur in 6 dimensions because the orientation must also be specified in addition to the position; our control system uses the same speed profile for both position and orientation.

The desired trajectory for testing the system is a straight line over the space at a given maximum linear velocity $\omega_{\max}$; hence, a trapezoidal velocity profile is utilized as interpolator

according to the acceleration equation:

$$\alpha(k) = \begin{cases} +a, & 0 \leq kT_s \leq \dfrac{T}{3}, \\ 0, & \dfrac{T}{3} < kT_s \leq \dfrac{2T}{3}, \\ -a, & \dfrac{2T}{3} < kT_s \leq T, \end{cases} \tag{3}$$

where $a > 0$ is a constant acceleration, T is the duration of the motion, and T_s is the sampling period. The motion period is computed as

$$T = \frac{3\theta_f}{2\omega_{\max}}, \tag{4}$$

where θ_f and $\omega_{\max}$ are the distance and maximum speed, respectively. On the other hand, the acceleration is obtained as

$$a = \frac{3\omega_{max}}{T}. \tag{5}$$

For the computation of the velocity function, the backward rectangular method (BRM) of numeric integration is used by setting

$$\omega(k) = \omega(k-1) + \alpha(k) \times T_s. \tag{6}$$

Analogously, the position at any given time $t = kTs$ is computed as

$$\theta(k) = \theta(k-1) + \omega(k) \times T_s. \tag{7}$$

An RT task within the kernel-space of the OS executes the computations for the six velocity profiles. These profiles describe dimensionless quantities; therefore, they may represent either angles or distances, it depends on the coordinate system used as described on Table 4.

TABLE 4. Corresponding trapezoidal velocity profile function for each coordinate system.

Profile	Joints	Cartesian
1	J1 angle	X coordinate
2	J2 angle	Y coordinate
3	J3 angle	Z coordinate
4	J4 angle	Roll angle
5	J5 angle	Yaw angle
6	J6 angle	Pitch angle

For Cartesian coordinates the first three velocity profiles describe the position coordinates while the last ones represent the orientation angles. Every single tool-center-point (TCP) $[p]$ and orientation matrix $[Q]$ is mapped to joint coordinates by solving the position inverse kinematics problem.

A set of coordinate frames F_i are attached to the robot joints as Fig. 9 shows, according to the Denavit-Hartenberg (DH) convention [30]; Table 5 enlists those parameters. We describe vectors $[\mathrm{a}_i]_i$ in the F_i frame which represent the coordinates of the origin for the F_{i+1} frame as

$$[\mathrm{a}_i]_i = \begin{bmatrix} a_i \cos\theta_i \\ a_i \sin\theta_i \\ b_i \end{bmatrix}. \tag{8}$$

TABLE 5. Denavit-Hartenberg parameters for the Fanuc S420F robot.

Joint	a(mm)	b(mm)	α(deg)
1	270.0	1000.0	90.0
2	900.0	0.0	0.0
3	270.0	0.0	90.0
4	0.0	1300.0	90.0
5	0.0	0.0	90.0
6	0.0	-270.0	0.0

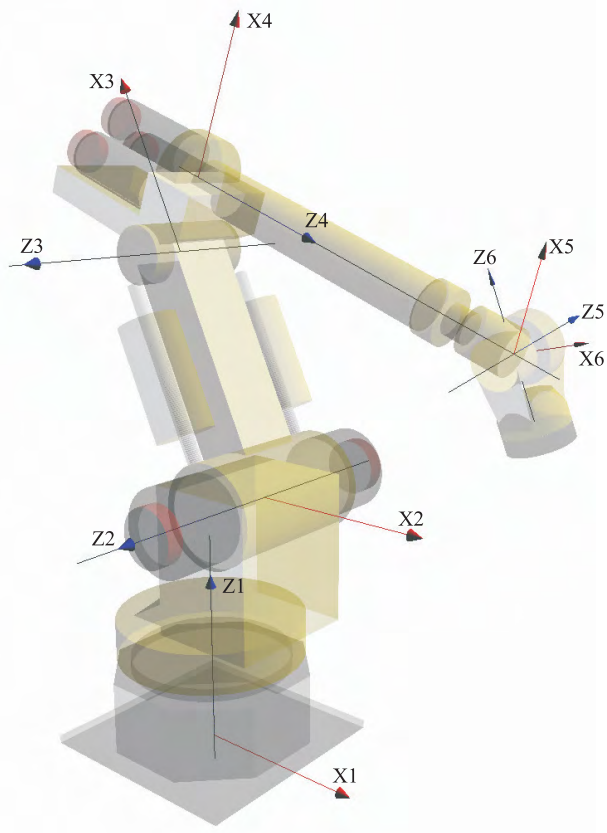


FIGURE 9. Coordinate frames attached to the robot joints.

In order to compute the forward kinematics it is necessary to perform a vector sum $\sum_{i=1}^{6} \mathrm{a}_i$; however, these vectors are described in different coordinate frames and, therefore, a homogeneous transform matrix

$$[Q_i]_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{9}$$

is used to map vectors from the $(i+1)$-th frame to the i-th frame.

By substituting the DH paramenters for each position vector above, the solution for the forward kinematics of position is obtained by computing equations

$$[c] = [\mathrm{a}_1]_1 + [Q_1]_1([\mathrm{a}_2]_2 + [Q_2]_2([\mathrm{a}_3]_3 + [Q_3]_3([\mathrm{a}_4]_4))) \tag{10}$$

and

$$[p] = [c] + \left(\prod_{i=1}^{5}[Q_i]_i\right)[a_6]_6, \tag{11}$$

where $[c]$ is the wrist center of the manipulator, while the attitude is given by

$$[Q] = \prod_{i=1}^{6}[Q_i]_i. \tag{12}$$

In the opposite direction, the complete solution for the inverse kinematics is computed as follows:

- Step 1. Compute wrist center $[c] = [p] - [Q][a_6]_7$, being $[c] = \begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T$ and $[a_6]_7 = \begin{bmatrix} 0 & 0 & b_6 \end{bmatrix}^T$.
- Step 2. $\theta_1 = \text{Arg}(X_c, Y_c)$.
- Step 3. $\sin\phi = \phi_n/(2.38994 \times 10^6)$ where $\phi_n = X_c^2 + Y_c^2 + (Z_c - b_1)^2 - 2\,a_1(c_1X_c + s_1Y_c) - 2.5 \times 10^6$, $c_1 = \cos\theta_1$ and $s_1 = \sin\theta_1$.
- Step 4. $\cos\phi = \pm\sqrt{1 - \sin^2\phi}$.
- Step 5. $\theta_3 = \text{Arg}(\cos\phi, \sin\phi) - 0.204781$, and let $c_3 = \cos\theta_3$ and $s_3 = \sin\theta_3$.
- Step 6. Solve the equation:

$$\begin{bmatrix} \cos\theta_2 \\ \sin\theta_2 \end{bmatrix} = \begin{bmatrix} a_2 + a_3c3 + b_4s3 & a_3s3 - b_4c3 \\ -a_3s3 + b_4c3 & a_2 + a_3c3 + b_4s3 \end{bmatrix}^{-1} \times \begin{bmatrix} c1X_c + s1Y_c - a_1 \\ Z_c - b_1 \end{bmatrix}.$$

- Step 7. Let $\theta_2 = \text{Arg}(\cos\theta_2, \sin\theta_2)$.
- Step 8. Using θ_1, θ_2, and θ_3, compute

$$[R] = [Q_3]_3^T[Q_2]_2^T[Q_1]_1^T[Q] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

- Step 9. Compute $s_5 = \sqrt{1 - r_{33}^2}$, $c_5 = -r_{33}$, and $\theta_5 = \text{Arg}(c_5, s_5)$, and
- Step 10. analogously, $s_4 = \text{sign}(\theta_5)r_{23}$, $c_4 = \text{sign}(\theta_5)r_{13}$, and $\theta_4 = \text{Arg}(c_4, s_4)$,
- Step 11. and finally, compute $s_6 = \text{sign}(\theta_5)r_{31}$, $c_6 = \text{sign}(\theta_5)r_{31}$, and $\theta_6 = \text{Arg}(c_6, s_6)$.

D. IMPLEMENTATION AND EXPERIMENTAL SETUP

The motion controller has been implemented in a Xilinx Spartan-3 XC3S1000 [31] device. The number of ocuppied slices is around 39% which corresponds to the logic of the position control for the six robot joints and the peripherals for data exchange, detailed above in Table 6. Special care has been taken given that this particular device has only twenty-four 18 × 18 multipliers. Two multipliers are used for the computation of the position control signal and another two for the commutation, being four multipliers per axis; thus, the design fulfills the multipliers capacity of the chip.

The Digilents Spartan-3 starter kit board includes a crystal oscillator of 50 MHz. On the other hand, the timing report from Table 6 suggest that the maximum clock frequency allowed is 82.143 MHz; therefore, timing constraints are successfully met.

TABLE 6. Synthesis report summary for the motion controller design.

Device utilization summary			
Logic utilization	Used	Available	Utilization
Number of slice flip-flops	2998	15360	19%
Number of 4 input LUTs	3743	15360	24%
Number of occupied slices	3048	7680	39%
Total number of 4 input LUTs	3792	15360	24%
Number of bonded IOBs	100	173	57%
Number of MULT18X18s	24	24	100%
Number of BUFGMUXs	1	8	12%
Timing summary			
Minimum period		12.174 ns	
Maximum frequency		82.143 MHz	
Minimum input arrival time		7.179 ns	
Maximum output required time		12.12 ns	
Maximum combinational path delay		11.299 ns	

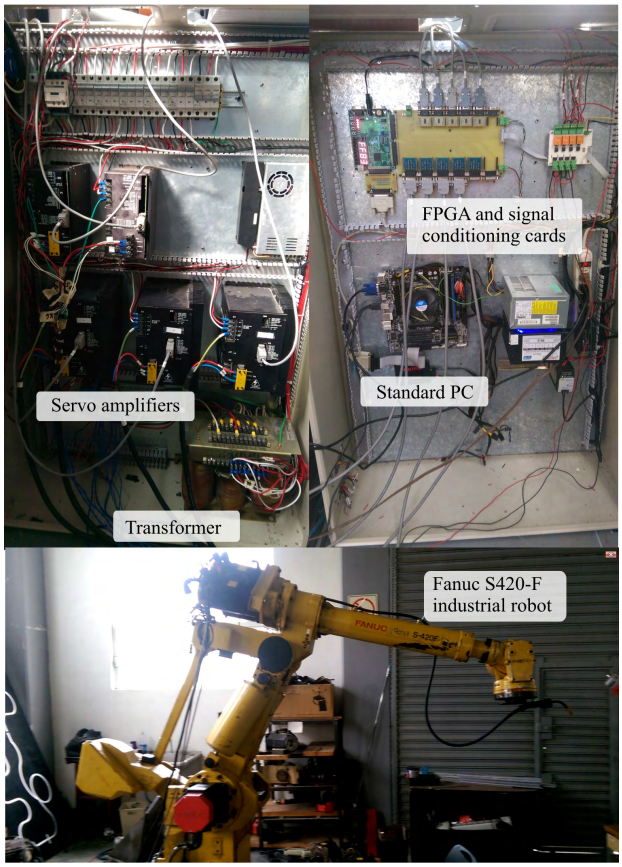


FIGURE 10. Experimental setup.

The experimental setup comprises an industrial Fanuc S420F robot (bottom of Fig. 10) actuated by six PMSM; each motor has an attached optical encoder for position and commutation feedback. The power cabinet (top-left corner of Fig. 10) has six AMC servo amplifiers with external commutation, three of them rated at 30A of continuous current for the arm joints and the other three rated at 15A for the wrist joints. The system is powered directly with a 3-phase line at 220V.

FIGURE 11. Main screen of user interface application software.

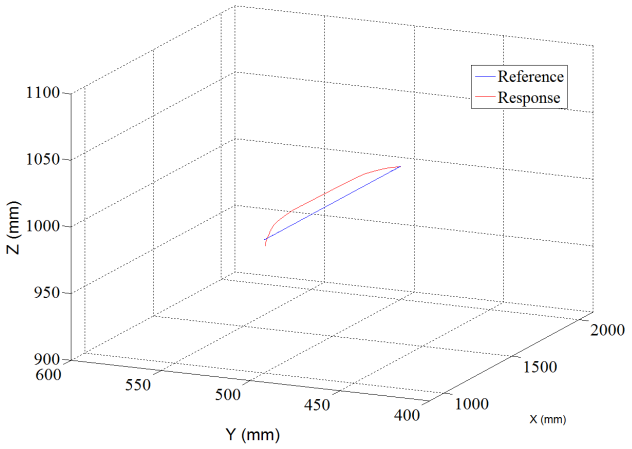


FIGURE 12. Cartesian response of the system at a feedrate of 500mm/s.

Given that the robot motors are rated at 150V, a transformer is used to step down the line voltage.

A Spartan-3 starter kit board is included in the control cabinet (top-right corner of Fig. 10). Several PCBs are connected to the FPGA for conditioning the commutation signals, electrical translation of the encoder signals, activation/deactivation of servo amplifiers, deactivation motor breaks, and for coupling the master interface signals.

As mentioned earlier, a standard desktop PC with an Intel® Core™ i7-4770 CPU at 3.40GHz is utilized to coordinate and to execute the robot trajectories. The PC runs the LTS Ubuntu 12.04 OS with the RTAI 3.6 patch. A Qt-based application, which is depicted in Fig. 11, has been developed to serve as a human-machine interface. This application has two main operation modes: manual and automatic; both modes perform movements either with joints or Cartesian coordinates.

IV. RESULTS AND DISCUSSION

Given that the purpose of this robotic system is the arc welding application, one of the tests of interest is a straight line trajectory in the Cartesian space. Thus, as a test, a point to point displacement using a linear interpolation is programmed into the software. The initial point has the coordinates $p_1 = \begin{bmatrix}2000 & 500 & 1000\end{bmatrix}^T$ while the final one is $p_2 = \begin{bmatrix}1000 & 500 & 1000\end{bmatrix}^T$, and the orientation for both points is given according to the Euler angles $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$ where the first angle represents a rotation about the Z axis, while the second and third ones are rotations about the Y and X axes, respectively.

The total displacement of the TCP along the X axis is $\theta_f = 1000$mm and the maximum velocity is set at $\omega_{\max} = 500$mm/s. Thus, by following a trapezoidal velocity profile, the period of the movement is $T = 3$s and the acceleration rate is $a = 500\text{mm/s}^2$.

The Cartesian response of the system is depicted in Fig. 12. The data series in this Fig. are collected from the profile generator output and from the encoder measurement every

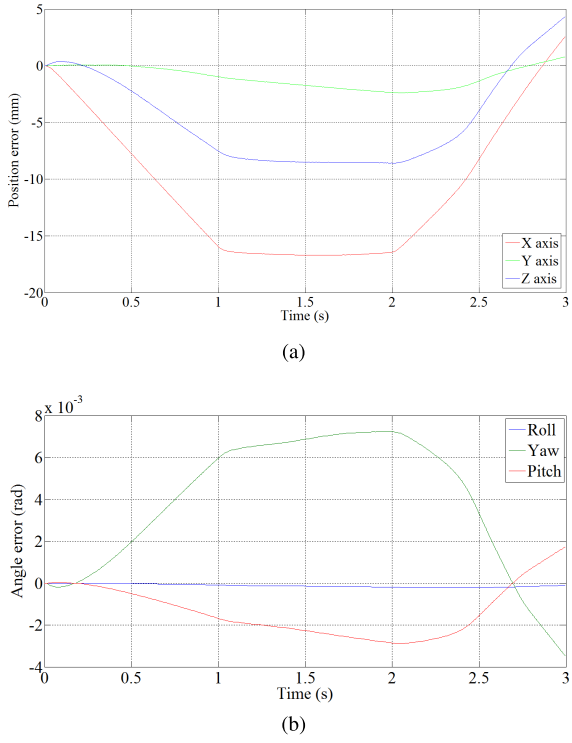
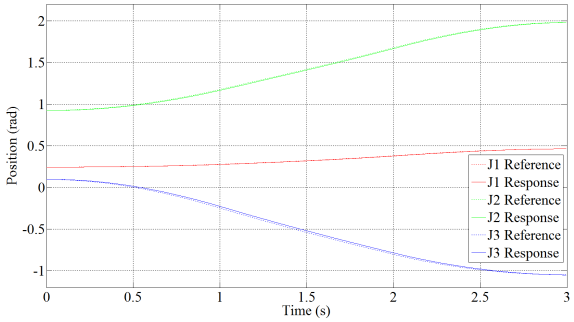


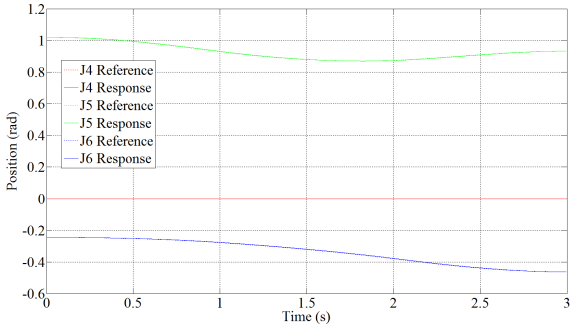
FIGURE 13. Tracking error: (a) position and (b) orientation.

millisecond; thus, we obtain three thousand data points for the test. The data obtained from incremental encoders is converted to joint angles, and then the forward kinematics solution is applied to each point in order to get the current Cartesian position of the robot.

The tracking RMS error for the position is illustrated in Fig. 13a. This curve represents the distance from the commanded position to the measured one, i.e., the norm of the error vector at a given time. The RMS tracking error is 13.7827, and the maximum position error is around 18.2425mm. Likewise, the orientation error is shown in Fig. 13b, where the three curves corresponding to the error for the three Euler angles are displayed.



(a)



(b)

FIGURE 14. System response at a feedrate of 500 mm/s. (a) Joints 1,2, and 3; (b) Joints 4,5, and 6.

Lastly, Figs. 14a and 14b depict the individual response for joints 1, 2, 3, 4, 5, and 6, respectively. Another important feature is that the joints 2 and 3 are decoupled, i.e., the motion of joint 2 does not alter the angle of joint 3; however, for the sake of simplicity of plots, the results are given as if those joints were coupled.

V. CONCLUSIONS

An open-architecture controller is proposed in this article; moreover, experimental data demonstrates that this architecture is completely functional and may be adopted as a starting point to develop a more sophisticated solution.

Every year a great number of robotic units are disposed from factories; hence, authors believe that a low-cost solution as the one presented in this paper may represent, for small and medium-sized industries, the opportunity of incorporating this sort of machinery to their processes and thus, increment their competitiveness.

The developed system achieves almost the same performance as the original controller, but this new version has major advantages such as modularity, given that it is possible to use H-bridges instead servo amplifiers, for example. In a similar manner, the resulting system is flexible in the sense that it may be implemented for a different robot architecture for instance, or simply, it is possible to connect a feedback device with a higher resolution without any modification. Besides, the usage of reconfigurable logic permits to change the control-law if an improved system is desired. Furthermore, the proposed system is scalable because the number of servo control instances may be adjusted according to the number of DoF of the robot to be controlled.

Furthermore, authors are convinced that the future of industrial controllers is aimed at the usage of programmable SoC, i.e., the implementation of software and hardware on a single silicon device. This may bring some benefits such as reducing system complexity, reduced time delays in data transfers between PC and FPGA, greater robustness to external electrical interference, and of course, higher cost savings.

REFERENCES

[1] T. Brogårdh, "Present and future robot control development—An industrial perspective," *Annu. Rev. Control*, vol. 31, no. 1, pp. 69–79, 2007.
[2] W. E. Ford, "What is an open architecture robot controller?" in *Proc. IEEE Int. Symp. Intell. Control*, Aug. 1994, pp. 27–32.
[3] F. M. Proctor and J. S. Albus, "Open architecture controllers," *IEEE Spectr.*, vol. 34, no. 6, pp. 60–64, Jun. 1997.
[4] G. C. Anzalone, C. Zhang, B. Wijnen, P. G. Sanders, and J. M. Pearce, "A low-cost open-source metal 3-D printer," *IEEE Access*, vol. 1, pp. 803–810, 2013.
[5] K.-S. Hong, K.-H. Choi, J.-G. Kim, and S. Lee, "A PC-based open robot control system: PC-ORC," *Robot. Comput.-Integr. Manuf.*, vol. 17, no. 4, pp. 355–365, Aug. 2001.
[6] Y.-S. Kung and G.-S. Shu, "Development of a FPGA-based motion control IC for robot arm," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Dec. 2005, pp. 1397–1402.
[7] J. Gu and C. W. de Silva, "Development and implementation of a real-time open-architecture control system for industrial robot systems," *Eng. Appl. Artif. Intell.*, vol. 17, no. 5, pp. 469–483, Aug. 2004.
[8] G. Ferretti, G. Magnani, P. Putz, and P. Rocco, "The structured design of an industrial robot controller," *Control Eng. Pract.*, vol. 4, no. 2, pp. 239–249, Feb. 1996.
[9] P. Putz and A. Elfving, "A development methodology for space A&R control systems," in *Automatic Control in Aerospace 1992*. Elsevier, 1993, pp. 363–368.
[10] A. Macchelli and C. Melchiorri, "A real-time control system for industrial robots and control applications based on real-time Linux," *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 55–60, 2002.
[11] D. Beal, E. Bianchi, L. Dozio, S. Hughes, P. Mantegazza, and S. Papacharalambous, "RTAI: Real-time application interface," *Linux J.*, vol. 29, no. 10, 2000.
[12] K. Nilsson and R. Johansson, "Integrated architecture for industrial robot programming and control," *Robot. Auto. Syst.*, vol. 29, no. 4, pp. 205–226, Dec. 1999.
[13] W. Sperling and P. Lutz, "Designing applications for an OSACA control," in *Proc. Int. Mech. Eng. Congr. Expo.*, 1997, pp. 16–21.
[14] Delta Tau Data Systems, Inc. Chatsworth, CA, USA. (2014). *Delta tau Data Systems, Inc*. [Online]. Available: http://www.deltatau.com/DT_IndexPage/index.aspx
[15] V. M. Becerra, C. N. J. Cage, W. S. Harwin, and P. M. Sharkey, "Hardware retrofit and computed torque control of a PUMA 560 robot updating an industrial manipulator," *IEEE Control Syst.*, vol. 24, no. 5, pp. 78–82, Oct. 2004.
[16] M. Farooq and D.-B. Wang, "Implementation of a new PC based controller for a PUMA robot," *J. Zhejiang Univ. Sci. A*, vol. 8, no. 12, pp. 1962–1970, 2007.
[17] A. Alabdo, J. Pérez, G. J. Garcia, J. Pomares, and F. Torres, "FPGA-based architecture for direct visual control robotic systems," *Mechatronics*, vol. 39, pp. 204–216, Nov. 2016.
[18] A. Visioli and G. Legnani, "On the trajectory tracking control of industrial SCARA robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 49, no. 1, pp. 224–232, Feb. 2002.
[19] T. N. Chang, B. Cheng, and P. Sriwilaijaroen, "Motion control firmware for high-speed robotic systems," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1713–1722, Oct. 2006.
[20] X. Shao and D. Sun, "Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance," *IEEE Trans Ind. Informat.*, vol. 3, no. 4, pp. 312–321, Nov. 2007.

[21] G. Gu, L. Zhu, Z. Xiong, and H. Ding, "Design of a distributed multiaxis motion control system using the IEEE-1394 bus," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4209–4218, Dec. 2010.
[22] R. C. Sampaio, J. M. S. T. Motta, and C. H. Llanos, "An FPGA-based controller design for a five degrees of freedom robot for repairing hydraulic turbine blades," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 39, no. 8, pp. 3121–3136, Aug. 2017.
[23] P. Liandong, H. Xinhan, and M. Arif, "A PC-based open architecture controller for robot," *Inf. Technol. J.*, vol. 3, no. 3, pp. 296–302, 2004.
[24] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.
[25] Y.-S. Kung, R.-F. Fung, and T.-Y. Tai, "Realization of a motion control IC for *X*-*Y* table based on novel FPGA technology," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 43–53, Jan. 2009.
[26] Advanced Motion Controls. Camarillo, CA, USA. (Apr. 2017) *Analog Servo Drive S60A40AC*. [Online]. Available: https://dpk3n3gg92jwt.cloudfront.net/domains/amc/pdf/AMC_Datasheet_S60A40AC.pdf
[27] S. Buzzetti, M. Capou, C. Guazzoni, A. Longoni, R. Mariani, and S. Moser, "High-speed FPGA-based pulse-height analyzer for high resolution X-ray spectroscopy," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 4, pp. 854–860, Aug. 2005.
[28] F. Garufi, F. Acernese, A. Boiano, R. De Rosa, R. Romano, and F. Barone, "A hybrid modular control and acquisition system," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 1, pp. 295–301, Feb. 2008.
[29] Y. F. Chan, M. Moallem, and W. Wang, "Design and implementation of modular FPGA-based PID controllers," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1898–1906, Aug. 2007.
[30] J. Angeles, *Fundamentals of Robotic Mechanical Systems*, vol. 2. Springer, 2002.
[31] *Spartan-3 FPGA family: Complete Data Sheet*, Xilinx, San Jose, CA, USA, Nov. 2005.

MIGUEL-ANGEL MARTíNEZ-PRADO received the B.S. degree in automation engineering, the M.S. degree in instrumentation and automatic control, and the Ph.D. degree in engineering. He has been a Professor/Researcher with the Autonomous University of Querétaro since 2008, where he teaches courses in servo mechanisms, robotics, and digital systems with reconfigurable logic, among others. He has participated in the development of various research projects related to the productive sector. His field of research includes motion control systems and industrial robotics.

JUVENAL RODRíGUEZ-RESÉNDIZ (SM'13) received the B.S. degree in automation engineering, the M.S. degree in instrumentation and automatic control, and the Ph.D. in engineering. He has been a Professor/Researcher with the Autonomous University of Querétaro (UAQ) since 2010. He is currently Coordinator of B.S. and M.S. in Automation at UAQ. His professional experience includes signal processing in software and hardware. He is President of the IEEE Querétaro section. He is a member of the National System of Researchers and the Mexican Academy of Sciences. He was a recipient of the Award from the Mexican Academy of Sciences in 2016.

ROBERTO-AUGUSTO GÓMEZ-LOENZO received the B.S. degree in applied mathematics, the M.S. degree in instrumentation and automatic control, and the Ph.D. degree in engineering. He has been a Professor/Researcher with the Autonomous University of Querétaro since 2004, where he has developed projects related to robotics, computers, and mathematics, and where he has taught several subjects. His main interest topics are control theory, digital systems, robotics, and computation theory.

GILBERTO HERRERA-RUIZ received the B.Sc. degree in electronics and the master's degree in electrical engineering from the Monterrey Institute of Technology and Higher Education, Mexico, and the Ph.D. degree in mechanical engineering from the Budapest University of Technology and Economics, in 1997. He is currently the Rector with the Autonomous University of Queretaro and he is a National Researcher level 3 with the Mexican Council of Science and Technology, CONACYT.

LUIS-ALFONSO FRANCO-GASCA received the B.Sc. degree in electronics engineering, the M.Sc. degree in electrical engineering, and the Ph.D. degree in mechatronics engineering. He was a Researcher with the Automatic Control and Dynamic Systems Laboratory, Advanced Technology Center, CIATEQ, Queretaro, Mexico, where he developed hardware for intelligent control applications and evolvable hardware. He currently designs, validates, and verifies hardware—as per DO254—for high performance aviation systems. His main research interests include digital signal processing, intelligent control algorithms, and FPGA embedded systems applications based on Heuristics techniques.