COBEM2015

ABCM Associação Brasileira de Engenharia e Ciências Mecânicas

23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

AN FPGA-BASED CONTROLLER DESIGN FOR A 5 D.O.F. ROBOT FOR REPAIRING HYDRAULIC TURBINE BLADES

Renato Coral Sampaio
José Maurício S. T. Motta
Carlos Humberto Llanos
University of Brasilia, Department of Mechanic and Mecatronics Engineering, Control and Automation Team (GRACO), Campus University Darcy Ribeiro, Brasilia - CEP 70910-900 - Phone Number: (55 61) 3107-3300
jmmotta@unb.br, llanos@unb.br, renatocoral@gmail.com

Abstract. *This paper presents the design of a FPGA-based embedded system for control and monitoring of a 5 d.o.f. spherical topology robot actuated by electric stepper and DC servo motors. The embedded robot controller has been developed using a hardware/software co-design approach in which a Terasic DE2-115 development kit, together with several hardware modules such as timer, JTAG, I/O, memory and RS-232, USB are connected via the Avalon bus. Additionally, several specific hardware modules have been designed in order to provide signals to the motor drivers and measure encoder signals. Also, a power-board has been designed to provide an opto-coupled connection to the first three motor drivers and their respective encoders. Finally, the control system also provides a RS-232 interface to the pan-tilt, which provides the last two degrees of freedom to the manipulator. The robot inverse kinematics was described in C language, running in the Nios II soft-processor, which receives spatial coordinate points from two processes running on a PC via an USB interface. These processes are responsible for processing the surface using specific image processing techniques based on a LASER sensor. Other process, running on a PC, generate the trajectories that the robot will follow during the welding task. The embedded software guarantees a suitable synchronization among all motors of the manipulator by using a Finite State Machine (FSM).*

Keywords: *Dedicated Robots, Embedded Systems, Robot Vision, Turbine Blade Repairing, Welding Robots, FPGA Applications*

1. INTRODUCTION

Robot development can be a challenge since each application has specific needs having its own topology, load demand, speed and precision requirements, among other characteristics. A robot controller has to read data from sensors, process the kinematic model and be able to command the actuators to perform the needed trajectories smoothly and/or bring the tool-tip to the desired position. The goals of this work are to develop a specialized welding robotic system for repairing the surface of hydraulic turbine blades eroded by cavitation pitting and/or cracked by cyclic loading, reducing human risks and increasing the efficiency of the process as described by Motta *et al.* (2014). Although some service robots have already been constructed for turbine blade repairing (Hazel B. and P., 2012; Chen *et al.*, 2008; Xiang *et al.*, 2010), this robot shows improvements to the previous, since it was designed to be completely automated, with little human action when in operation. In addition, this robot can operate inside large turbines, with 8 m in diameter and at least 500 mm in between blades (Motta *et al.*, 2010a,b, 2014). The robot requirements include being able to scan the 3D blade surface to be repaired, detecting and creating a virtual model of the area to be filled by metal, creating a welding strategy involving trajectory and welding parameters and finally controlling the robot and the welding power source to execute the welding process. Also, the robot has to support the welding torch, be light to guarantee portability and have an average accuracy of a least half the diameter of the welding wire (1.2 mm) to be able to perform a high quality welding.

For the controller platform specification there are many available options ranging from GPPs (General Purpose Processors) to DSPs (Digital Signal Processors), from reconfigurable dedicated hardware designs based on FPGAs (Field Programmable Gate Arrays) to dedicated hardware solutions such as ASICs (Application Specific Integrated Circuits). In order to be able to embed as much intelligence as possible in our controller, while including as few as possible separate processing units, an FPGA platform was chosen so that it can embed, in one chip, many parallel systems to process not only all kinematics and motor control algorithms, but also communication protocols, and in a future stage of the project, include real-time image processing from the camera of the 3D Laser Scanner. As a flexible platform, an FPGA offers the ability to create a custom architecture, in which it is possible to use a co-design approach, for instance embedding a GPP for general tasks along with dedicated hardware modules running in parallel in order to accomplish specific critical tasks. Additionally, an embedded system design based on FPGA provides a rapid system prototyping mean given the flexibility of these devices for complex systems design, allowing the reduction of TTMs (Time-to-Market) of new technical

developments because of shorter time required for design, debugging and test processes.

Following previous publications related to this project in (Ginani and Motta, 2007), (Sánchez *et al.*, 2010) and (Motta *et al.*, 2010b), this article focuses on the system architecture that controls all the different modules that make the robot work. Starting in section 2.the overall system description is presented followed by section 3.which describes the controller architecture in the FPGA including the embedded kinematic algorithms and dedicated hardware modules. Section 4.briefly describes the software modules running on the PC. Section 5.presents the design of the interface boards and peripheral devices and finally section 6.presents the conclusion.

2. THE SYSTEM DESCRIPTION

As described by the system requirements the system has to accomplish many diverse tasks. In this case, the overall architecture is composed of 6 main modules that work together to achieve this goal shown in Figure 1. The central module is the *Robot with Welding Torch* (A) which is a 5 d.o.f. spherical robot powered by three step motors (in the manipulator) and two DC Motors (Pan-Tilt). The *3D Laser Scanner* (B) is attached to the robot arm and controlled via PC through a USB cable. The *MIG/MAG Power Source* (C) feeds the metal wire and current/gas being controlled by the *ROB 5000* module. The *FPGA Embedded Robot Controller* (D) is responsible for receiving the trajectories from the PC via USB and commanding both the robot motors and the MIG/MAG Power Source to execute the welding process. The *PC* (E) runs both the *3D Laser Scanner Software* and the *Welding Trajectory Planner*. Finally, the *joystick* with its controller (F) is used to manually move the robot. An important aspect of this system is that each module needs to be integrated with at least another one, having its own communication protocol, in order to achieve the communication demands. Therefore, hardware and software interfaces had to be built. These interfaces and the main control algorithms are described in the next sections.

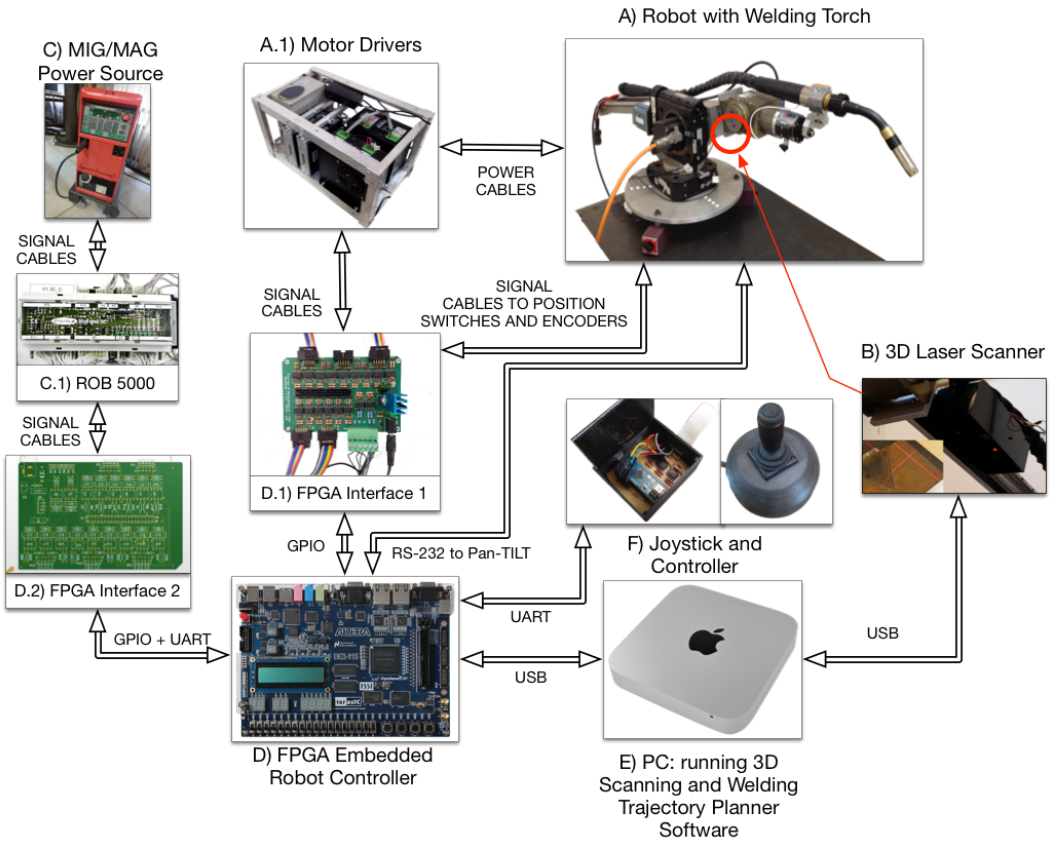


Figure 1: The overall Robot Control System

3. THE FPGA EMBEDDED CONTROLLER

The central robot controller is designed on a Reconfigurable Architecture based on an FPGA. The same has been developed by using a hardware/software co-design approach, in which a soft-processor, the NIOS II (Altera, 2011), has been embedded in an Altera Cyclone IV FPGA on the DE2-115 development kit, together with several standard hardware modules such as timer, JTAG, I/O, UART, Memory and RS-232, ethernet, and connected via the Avalon bus. Additionally, three specific hardware modules have been designed (namely, *Step Motor Controllers*, *Encoder Counters* circuits and *USB-Module*) providing: (a) signals to the motor drivers, (b) a measurement mean for signals from encoders, and (c) a

communication channel with the PC.

Figure 2 describes the FPGA system architecture along with all the peripheral components it connects to. The central piece of the architecture is a 32 bit Nios II soft-processor running at 100 MHz with a hardware Floating-Point Unit. The processor acts as the master of the Avalon BUS through which it connects to all peripherals. The main memory module is an SSRAM of 2MB. Additionally there are modules that use parallel communication (PIO - Parallel I/O) to connect to the processor as well as modules that use serial communication which is handled independently using the RS232, UART1, UART2 and USB modules.

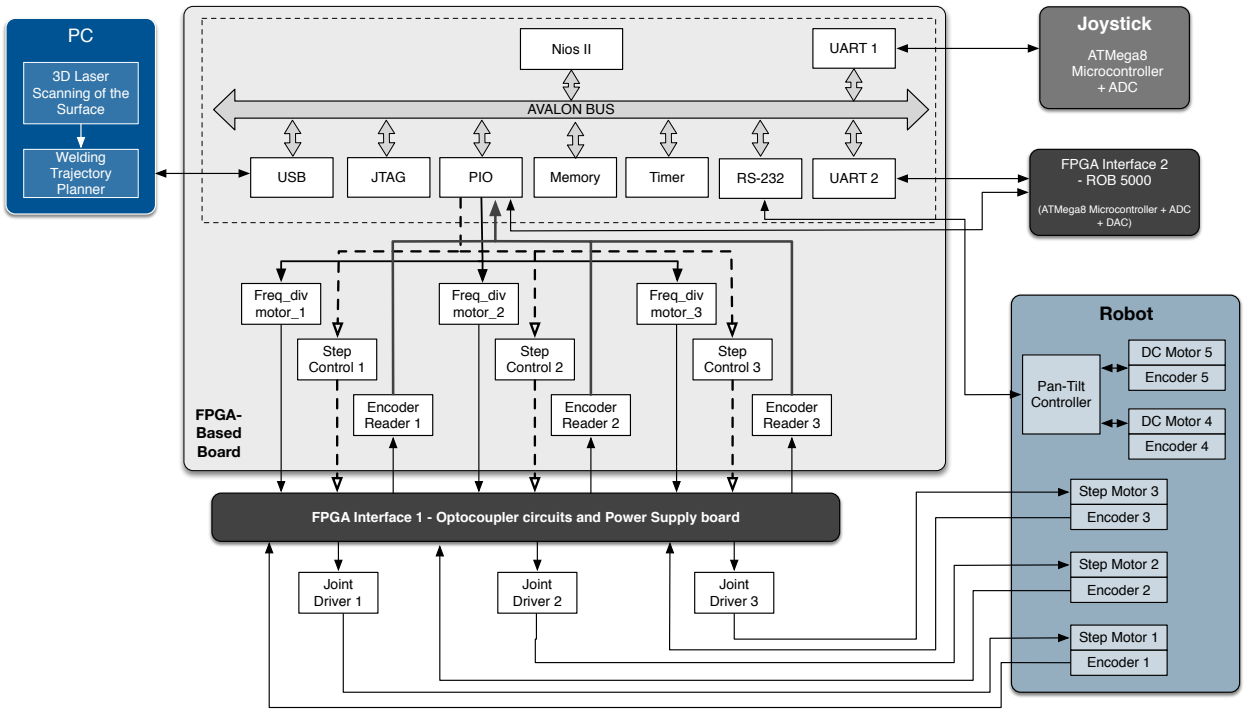


Figure 2: System Architecture Diagram

The PIO module is used to connect to both internal dedicated hardware components and external circuitry on the interface boards. The dedicated hardware components are the *Step Motor Controllers*, the *Encoder Counters* and an *USB module*. There are three *Step Motor Controllers* that receive the number of steps, direction and frequency in order to independently control the step motors. The processor only needs to inform this data once and wait for the hardware module to control the movement. Also, there are three independent *Encoder Counters* recording the number of steps for each motor, allowing the processor to read their values whenever it needs without being interrupted. The external circuitry includes the *FPGA Interface Boards* 1 and 2 that use PIO to connect to the *Step Motor Controllers*, encoders, hall effect sensors and the *ROB 5000 controller* (in this case for critical signals such as *emergency stop* and *collision detection*). The serial communication components are the following: (a) the RS-232 serial port used to control the Schunk PW-90 Pan-Tilt which has its own set of commands that were implemented via software; (b) the UART 1 used to communicate with the joystick, which is controlled by an ATMega8 Microcontroller; (c) the UART 2 that communicates with the *FPGA Interface 2 board* sending and receiving commands to the welding power source through the ROB 5000 controller; and (d) the USB Module used to receive commands from the PC and send the current state of the robot such as task and position.

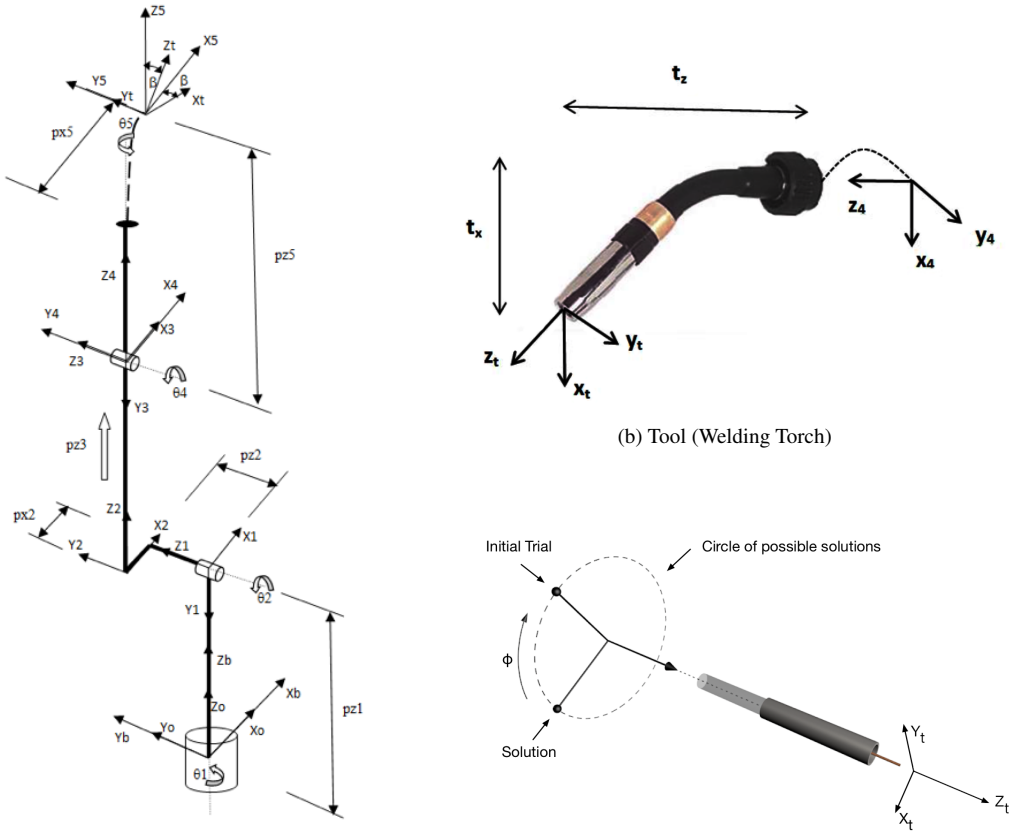
3.1 The Embedded Software

The embedded software running on the Nios II is written in C being structured to make the robot work as both a stand-alone device or as a host that receives commands via USB from the PC. It has some initializing tasks such as moving the robot to a home position and then operates on a loop-based algorithm that either is controlled by the joystick or waits for the PC to issue commands. The basic commands were defined to execute a *Point-to-Point movement*, a *Straight-Line movement* or to execute a more complex trajectory, combining the above movements and the welding control. More advanced commands include calibration and manual movement directly from the FPGA kit.

3.2 Kinematics Algorithms

3.2.1 Forward Kinematic Model

For the construction of the forward kinematic model, joint coordinate systems have been assigned in such a way that the Z_n axes point towards the joint axis of motion and that the vectors X_n are all parallel when the robot is in its



(a) Robot

(b) Tool (Welding Torch)

(c) Iterative search of the third orientation angle, Φ

Figure 3: Robot Diagram

zero position. The Y_n axes are orthogonal to the formers, creating an orthonormal systems of axes. This follows the Denavit_Hartemberg (D-H) convention as described in (McKerrow, 1991) and (Paul, 1981).

Homogeneous transformation matrices relating coordinate frames from the base (b) to the torch/tool (t) can be derived as defined in Eq. 1:

$$ {}^{b}_{t}T = {}^{b}_{0}T * {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T * {}^{4}_{5}T * {}^{5}_{t}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1} $$

where ${}^{i}_{i+1}T$ is the homogeneous transformation between two successive joint coordinate frames. The welding torch is attached to link 4 and its geometric parameters are shown in Figure 3.

$$ {}^{i-1}_{i}T = Rz(\theta) * Tz(d) * Tx(l) * Rx(\alpha) = \begin{bmatrix} \cos(\theta) & -\cos(\alpha) * \sin(\theta) & \sin(\alpha) * \sin(\theta) & l * \cos(\theta) \\ \sin(\theta) & \cos(\alpha) * \cos(\theta) & -\cos(\theta) * \sin(\alpha) & l * \sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2} $$

Using the geometric parameters shown in Fig. 3 and applying them for each robot joint in Eq. 2, and then multiplying every joint transformation such as in Eq. 1 then the robot kinematic model can be described in a matricial form as described in Equations 3 and 4.

$$ {}^{b}_{t}T = \begin{bmatrix} -c_b.(s_1.s_5 - c_5.c_1.c_{24}) - s_b.c_1.s_{24} & -s_5.c1.c_{24} - c_5.s_1 & c_b.c_1.s_{24} - s_b.(s_1.s_5 - c_5.c_1.c_{24}) & p_x \\ c_b.(c_1.s_5 + c_5.s_1.c_{24}) - s_b.s_1.s_{24} & c_1.c_5 - s_5.s_1.c_{24} & s_b.(c_1.s_5 + c_5.s_1.c_{24}) + c_b.s_1.s_{24} & p_y \\ -s_b.c_{24} - c_b.c_5.s_{24} & s_5.s_{24} & c_b.c_{24} - s_b.c_5.s_{24} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} pz5.c_1.s_{24} + tz.c_1.s_{24} - pz2.s_1 - tx.(s_1.s_5 - c_5.c_1.c_{24}) + pz3.c_1.s_2 - px2.c_1.c_2 \\ pz5.s_1.s_{24} + tz.s_1.s_{24} + pz2.c_1 + tx.(c_1.s_5 + c_5.s_1.c_{24}) + pz3.s_1.s_2 - px2.c_2.s_1 \\ pz1 + pz5.c_{24} + tz.c_{24} + pz3.c_2 + px2.s_2 - tx.c_5.s_{24} \end{bmatrix}, \quad (4)$$

where $s_1 = \sin(\theta 1)$, $c_2 = \cos(\theta 2)$, $s_{24} = \sin(\theta 2 + \theta 4)$ and $c_b = \cos(\beta)$.

3.2.2 Inverse Kinematic Model

As discussed in (Motta *et al.*, 2014), a robot with 5 d.o.f. cannot locate its end-effector/tool at every pose within its workspace, which means that one of the three orientation angles cannot be chosen freely with a 2 d.o.f pan-tilt wrist. Therefore, instead of using a sextuple of 3 position and 3 orientation coordinates a solution is achieved by using the triplet for position and only two orientation angles. The third angle is a function of the other five pose parameters and the robot joint variables. Since this limitation impedes that explicit equations are found a recursive algorithm is used to choose an initial value for the third orientation angle and solve the equations with as few iterations as possible. This approach uses the forward kinematic model to calculate the difference between the third orientation angle initial trial and the current one. In this process the error/residual is calculated from the difference between two calculated consecutive positions of the torch tip (vector *P* in Eq. 4). Including the tool frame, the angle Φ in Fig. 3 (c) is the angle calculated between the vectors of the first and second columns in the matrix expressed in Eq. 1.

In practice, the inverse kinematics algorithm receives the position coordinate point $P = [Px, Py, Pz]$ and the desired orientation $\Omega_t = [X_t, Y_t, Z_t]$ unit vector, which is a normal vector to the surface to be welded, and it iteratively calculates the possible joint variables until the error is less than 0.2 mm.

3.2.3 Kinematic Algorithm Implementation

The implementation of the algorithm has to achieve a fast performance to be able to control the motors for a smooth trajectory. In this case, the welding linear speeds are approximately 10 mm/s and although response time of the step motor is under 10 ms (for a single step), a smooth trajectory can be guaranteed with a time interval of 100 ms between intermediate points. For a real-time controller all computational time has to be under 100 ms, and hence an optimization strategy has to be developed. As an example, the computational requirements for calculating the forward kinematics (see in Eq. 1) demands calculation of sine and cosine of 5 different angles ($\theta 1, \theta 2, \theta 4, \theta 5, \beta$), and also the multiplication of seven 4x4 matrices. In a simple *for* loop, this task requires 384 float-point multiplications and 288 float-point additions/subtractions beyond the sine and cosine evaluation. To reduce the calculation time, the resulting polynomial equations of the final matrix element is used as shown in Eq. 3 and Eq. 4. That reduces the number of multiplications and additions/subtractions to 69 and 32, respectively. Further optimization is carried out by identifying all repeated elements of those expressions and performing calculations only once per element. An example is the expression $s_1.s_5 - c_5.c_1.c_{24}$ that repeats itself three times on all equations. The final result is 45 multiplications and 30 addition/subtraction operations. This yields a speed up in performance of 9 times with respect to the original expression. In this case, the overall inverse kinematics algorithm takes about 20 ms to calculate each position, which is low enough to attend the time requirements of the robot.

3.3 The Point-to-Point movement

In the Point-to-Point movement the controller receives a destination pose (namely, *position* and *orientation* of the *end-effector*), as well as a time to execute a movement in which the intermediary positions along the trajectory are not relevant. In this case, controller uses the current pose of the robot and calculates the joint variables for the final positions using the Inverse Kinematics algorithm, as described in (Motta *et al.*, 2014). Afterwards, the angular difference of each joint is computed to define (a) the number of steps, (b) direction and (c) speed needed for each motor to accomplish the movement. In this case, the speed calculation of the joint movement is defined from the maximum speed allowed in each axis. The joint with the highest displacement will be that with higher speed and the remaining ones will have proportionally slower speeds to achieve a synchronized movement. In the pan-tilt case, a displacement function is used where each joint receives the absolute angle as well as the overall duration of the movement. The overall algorithm is shown in Fig. 4.

With the number of steps, direction and speed, the control algorithm is called to perform the movement which evaluates each joint movement and creates a trapezoidal shaped acceleration and a deceleration curve for the each motor, in order to smoothly perform the movement. At the end of each movement, the algorithm takes into account the encoder data for the first 3 axes and calculates the difference between the number of steps sent to the motor and the real desired position. If any difference is detected a correction movement is performed. Figure 5 (a) depicts the Point-to-Point movement from point A to B in which the achieved trajectory is an arc, following the base edge.

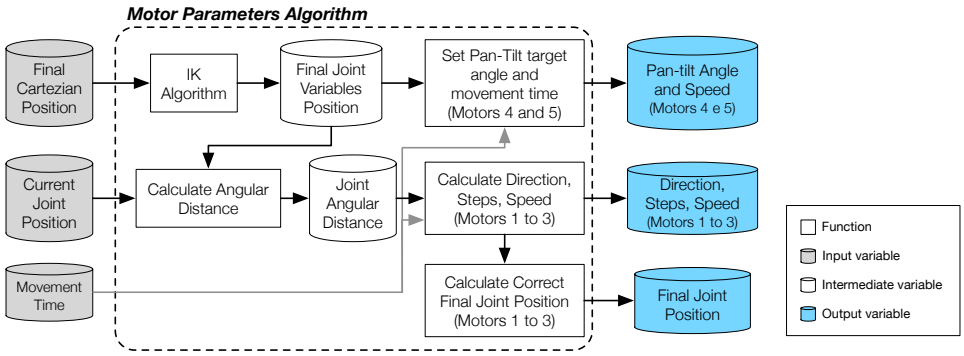
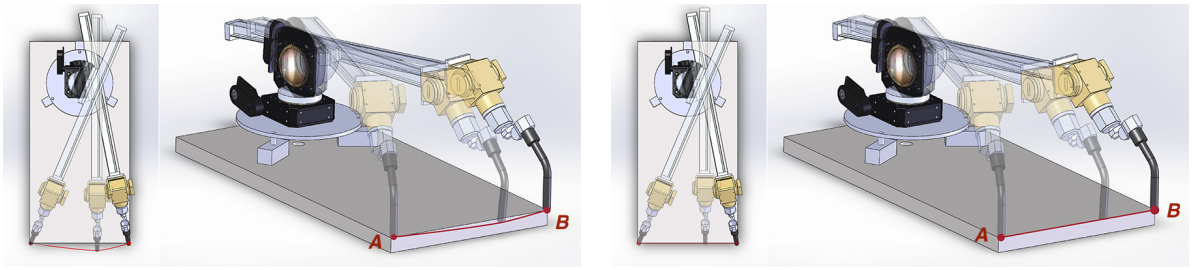


Figure 4: Point-to-Point Movement Algorithm Diagram



(a) Point-to-Point Movement (b) Straight Line Movement

Figure 5: Types of robot movements

3.4 The Straight-line example

The second type of movement implemented in the embedded system is related to straight lines where the initial and final points are defined and the algorithm creates an interpolation procedure in order to achieve a straight line in 3D space, guaranteeing the initial and final orientations being interpolated through a smooth movement. The algorithm, shown in Fig 6 consists of calculating the Euclidean distance between initial and final points and, from the desired spatial speed and time interval between points, defining all the intermediate points from small increments (0.5 mm) in order to create a linear trajectory. For each intermediate point, the number of steps, direction and speed are calculated using the *Motor Parameters Algorithm* show in the previous Point-to-Point movement.

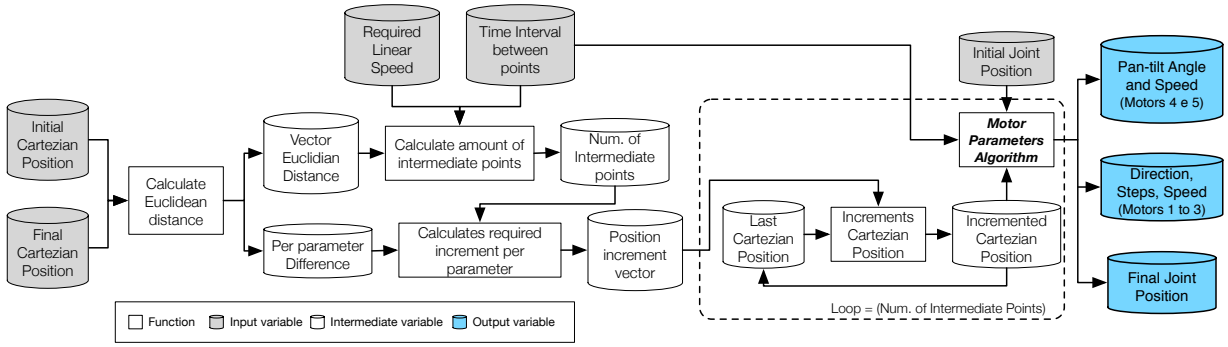


Figure 6: Straight Line Movement Algorithm Diagram

To achieve a straight line displacement the movement is decomposed on small straight-line segments, in which a correction process is carried out taking into account the encoder data. In this case, the number of steps is rectified adding or subtracting lost steps in the next straight-line segment to be executed. Figure 5 (b) depicts a movement between two points where the trajectory executed is a straight-line which is yielded from the interpolation among several intermediate points.

3.5 Joystick Movement

The *joystick mode* was implemented in order to provide the robot operator means to manually move and record desired positions for reference points, calibration or trajectory planning. Using the 3-axis joystick the operator can move the end-effector in X, Y or Z or change the orientation angle in each axis.

3.6 Trajectory Movement

The more complex operation mode is the trajectory movement, where a combination of movements and welding control can be performed. This trajectory is transmitted from the PC to the FPGA board. In this case, the possible commands include *Point-to-Point* movement, *Straight-Line* movement, *Welding Control* (init, stop, current, etc) and a *Wait* command.

3.7 The USB Module

The USB module was designed as a hardware module that operates in parallel to the processor. It is connected to the Avalon BUS via a wrapper. Beyond implementing the USB protocol, this module has two internal memory areas. One for input and the other for output data. Both memories operate independently so the module can be receiving data from the PC via USB to the *data in memory*, while also receives output data from the Nios II processor. A hardware logic controller sets flags for each memory so both the Nios II processor and the PC know when there is data to be read, and when there is available space to write new data. Figure 7 depicts the diagram for the USB hardware module.

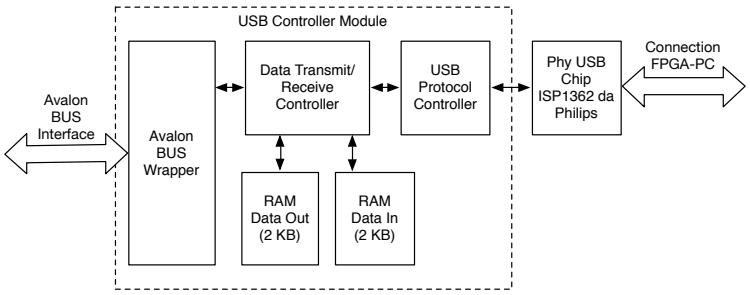


Figure 7: USB Hardware Module Diagram

4. THE SOFTWARE APPLICATIONS ON THE PC

On the PC, there are two applications running. The first one is the 3D Laser Scanner software which controls the scanner mechanism. It captures and processes the images from the camera sensor, then generates a 3D virtual model of the area to be filled with metal (Pizo, 2014). The second software is the *Welding Trajectory Planner* which reads the 3D model of the area and generates a welding trajectory according to the strategy selected, translating its output to the robot coordinate system, allowing the manipulator to perform the welding task (Pinheiro, 2014). For both applications there is a USB driver that was developed to enable direct communication and control the robot. Through this driver, the software on the PC is able to send commands to the robot such as simple *Point-to-Point* or *Straight-Line* movement commands to full trajectory execution. And since the USB module is running in parallel on the FPGA, the PC can also monitor the robot position in real-time at any moment.

5. DESING OF INTERFACE BOARDS AND PERIPHERAL DEVICES

5.1 The interface boards

The *FPGA Interface 1* board has been designed to provide an opto-coupled connection to the first three motor drivers and their respective encoders, together with power supplies to generate 5, 12 and 24 dc-voltages necessary to interface with each sensor. The same board provides appropriated connectors for achieving a robust and secure interconnections with the 3 steeper-motors. The optocouplers where chosen to protect the low voltage (3.3 VDC) of the FPGA chip. All pins are directly connected to the FPGA via GPIO in parallel mode. The *FPGA Interface 2* board is more elaborated than the previous one and was designed to connect the FPGA kit to the ROB 5000 controller. It also operates based on opto-couplers to convert 3.3 VDC to 5 and 24 VDC, but since the ROB 5000 also has analog inputs and outputs it incorporates two ATMega8 Microcontrollers, two MCP4922 DAC ICs and op-amps in order to implement the 5 analog inputs and 3 analog outputs. In this case, most commands are sent via serial UART protocol, since the time required to configure the welding power source is not an issue. The only direct parallel ports used are those for the emergency signals: the *Quick Stop* that cuts the welding power immediately, and the *Collision Detection* that sends a collision signal to the robot controller to stop the movement.

5.2 Joystick module

Finally, the *joystick module* was built providing a way to manually control the robot, enabling manual trajectory creation, reference point checking and calibration task. The joystick used is a pure analog 3-axis and a push button.

To interface it with the FPGA it required the build of a third small board containing optocouplers and an ATMega8 microcontroller that provides the ADC converters. As said earlier, the communication with the FPGA is accomplished via UART protocol and is connected to the expansion header on the board. The module can be seen in Fig. 1 (F).

6. CONCLUSION

This paper presented a research project of a welding robot that is currently in the final stages of development. The focus of this work was the main robot controller implementation. As described, the system architecture includes 6 main modules, each having its own local processing unit and its own communication protocol. The central control unit of the robot was implemented on a FPGA kit, running an embedded processor along with dedicated hardware modules. The integration of all modules required both software and hardware solutions to implement both the communication protocols logic and electronic interface boards.

The control algorithms for forward and inverse kinematics were optimized for fast performance on the embedded processor, being able to preform real-time movements acting as either a host device, executing commands sent from the PC software, or in independent mode, receiving commands from the joystick.

7. ACKNOWLEDGEMENTS

This work had been supported by ELETRONORTE (Electrical Power Plants of the North of Brazil) and FINATEC (Foundation for Scientific and Technological Enterprises).

8. REFERENCES

Altera, C., 2011. *Nios II Processor Reference Handbook*. Altera Corporation, San Jose, CA.

Chen, Q., Sun, Z., Zhang, W. and Gui, Z., 2008. "A robot for welding repair of hydraulic turbine blade". In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*. pp. 155–159.

Ginani, L. and Motta, J.M.S.T., 2007. "A laser scanning system for 3d modeling of industrial objects based on computer vision". *Proceedings of 19th International Congress of Mechanical Engineering*, Vol. 1, p. 10.

Hazel B., Coté J., L.Y. and P., M., 2012. "A portable, multiprocess, track-based robot for in situ work on hydropower equipment". *Journal of Field Robotics*, Vol. 29, pp. 69–101.

McKerrow, P., 1991. *Introduction to Robotics*. ACM Press frontier series. Addison-Wesley Publishing Company.

Motta, J., Llanos, C., Carvalho, G., Absi-Alfaro, S., Idrobo-Pizo, G. and Sampaio, R., 2014. "A specialized robotic system prototype for repairing surface profiles of hydraulic turbine blades". In *Applied Robotics for the Power Industry (CARPI), 2014 3rd International Conference on*. pp. 1–5.

Motta, J., Llanos, C., Carvalho, G. and Alfaro, S., 2010a. "A prototype of a specialized robotic system for repairing hydraulic turbine blades". In *Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on*. pp. 1–6.

Motta, J.M.S.T., de Carvalho, G.C., Quintero, C.H.L., de Britto Vidal Filho, W., Filho, E.P.V. and Ginani, L.S., 2010b. "A specialized welding robot for repairing hydraulic turbine blades". *ABCM Symposium Series in Mechatronics*, Vol. 4, pp. 700–709.

Paul, R., 1981. *Robot Manipulators: Mathematics, Programming, and Control : the Computer Control of Robot Manipulators*. Artificial Intelligence Series. MIT Press.

Pinheiro, L.S., 2014. *Mapeamento 3D e planejamento de trajetórias para preenchimento de cavidades por meio de soldagem em múltiplos passes usando o processo Gmaw robotizado*. Master's thesis, Universidade de Brasilia, Brasilia, DF.

Pizo, G.A.I., 2014. *Projeto de um descritor para o alinhamento de imagens de profundidade de superfícies com aplicação em visão robótica*. Ph.D. thesis, Universidade de Brasilia, Brasilia, DF.

Sánchez, D.F., noz, D.M.M., Llanos, C.H. and Motta, J.M., 2010. "A reconfigurable system approach to the direct kinematics of a 5 d.o.f robotic manipulator". *International Journal of Reconfigurable Computing*, Vol. 2010, p. 10.

Xiang, K., Sun, Z., Dai, H., Chen, Q. and Liu, J., 2010. "Can-bus based distributed control system for hydraulic turbine blade repairing robot". In H. Liu, H. Ding, Z. Xiong and X. Zhu, eds., *Intelligent Robotics and Applications*, Springer Berlin Heidelberg, Vol. 6425 of *Lecture Notes in Computer Science*, pp. 695–704.

9. RESPONSIBILITY NOTICE

The three authors are the only responsible for the printed material included in this paper.