The 4th TSME International Conference on Mechanical Engineering
16-18 October 2013, Pattaya, Chonburi
TSME-ICoME 2013
DRC-1006

A Reconfigurable Open Architecture Controller for a Research Robot

Kitithorn Phanatamporn[1] and Ratchatin Chancharoen*

[1] Chulalongkorn University, 254 Phayathai Road, Pathumwan, Bangkok, 10330
*Corresponding Author: Ratchatin.C@chula.ac.th, 662-2186643, 662-2522889

Abstract

Industrial Robot plays an important role in modern industry where products are very sophisticated and difficult to manufacture while environment and cost are also concerned. Its role becomes increasing important towards the future where robots are working closely with human in our living space. This paper proposes a reconfigurable open architecture controller for a robot in order to effectively control both its motion and force using standard interface devices such as a manual pulse generator and a CCD camera. The proposed controller is flexible that can control the robot in various modes and can interface with various robot sensors such as force and vision as well. These features are keys to develop a human friendly robot that works closely and safely with human using standard parts available in the markets. The presented controller is a PC-based robot controller that combines 3D printer software, i.e., Repetier and Slic3r with the powerful Matlab and Visual C++ to develop a full function robot controller. To demonstrate the performance of the controller, it is used to control a Delta robot to draw along a predefined trajectory on a part which its position is not a priori known.

Keywords: Open Architecture, PC-based robot controller

1. Introduction

An Industrial robot [1] is designed with precision motion control capability that is effectively used for transferring, inspection, and processing parts in industrial plant and plays an important role in modern manufacturing facility since 1970 to present [2]. Although the capability of an industrial robot is very interesting, its working performance is only achieved in the environment that is suit the robot. This robot should not be used in non-industrial domain such as in a hospital or an office where working space contains unstructured, dynamic environment and sometimes human.

The design of a non-industrial robot is very challenging since this robot requires sensors to sense object and/or environment in real time, human-friendly interface devices, and the advanced control technique to effectively control both force and motion [2]. This robot must be 1) safe for the environment, human and the robot itself and 2) intelligent such that it can still work effectively in dynamic environment. The commercial robot controller is normally closed architecture which its capability is limited [3, 4].

The project is to develop an open architecture controller [5] for a research robot to be a solution of these problems [4, 6, 7] such that we can use the robot's manipulator effectively in non-industrial domain using newly available and modern input devices and sensors. For examples, the controller is able to interface with various input devices

including Microsoft Kinect, Sony PlayStation Eye, Logitech Gamepad, manual pulse generator, 3Dconnexion SpaceMouse, Sensible haptic device and various sensors including JR3 force sensor and Hokuyo Laser Rangefinder and many devices. The controller is designed to effectively handle heavy mathematics and logic including robot kinematics and control algorithm in order to do complicated tasks that effectively control both the motion and force.

The proposed system consists of two computers; one is running a real-time code at a high servoing rate (1 kHz) and the other is running a Windows based program where programming's environment is excellent and can interface with devices through Window's driver. There are three updated rates including hardwiring for safety, 1 kHz for real time control and about 20-100 Hz for non-real time updated signals and graphic user interface. The control program is developed in Graphical language on Matlab/Simulink in order to utilize the mathematics from Matlab and to convert the program into real-time machine code via Matlab's xPC toolbox and MS visual C++. The Repetier [8] and Slic3r [9], a 3D printing program running on windows PC, is used to generate a G-code [10] from .stl file to control the robot in real time. In this way, the robot can be operated with G-code and/or the input device such as manual pulse generator (MPG).

The preliminary experiment is used to demonstrate some capabilities of the proposed controller, i.e., a delta robot is first controlled to position itself on the part that is not rigidly fixed to the base by the Manual pulse generator and using the digital camera as a feedback. Once the robot is in position, as seen by a camera, the robot draws a trajectory, which is written in G-code, on a paper.

2. Features of the Reconfigurable Open Architecture Controller

There are many concepts to define Open Architecture Controller. One preferred property of this controller is that it should be able to integrate with many modules and devices to improve flexibility, reliability, reconfigurability [11] and quality of work as well.

Our proposed system consists of two computers, Windows based and real-time computers which can communicate and exchange data between two computers via TCP/UDP. The external devices can connect to the Windows based computer via USB and to the real-time computer via PCI bus (Fig. 1). In this way, most devices in the market can interface with the system.

The other property is that it is reconfigurable mathematics. Normally, the robot controller must be able to handle the robot mathematics, including forward and inverse kinematic, Jacobian [1] and the low level PID controller, as a minimum. The proposed control diagram (Fig. 2) is designed such that it is also reconfigurable, to develop and test the new ideas. For example, the proposed controller can be easily reconfigured such that the reference trajectory is constructed from two input devices, i.e., Repetier and MPG (Fig 3.). The *X*, *Y* positions from Repetier are merged with the *Z* position from MPG such that robot goes automatically in plane while an operator control the depth manually.

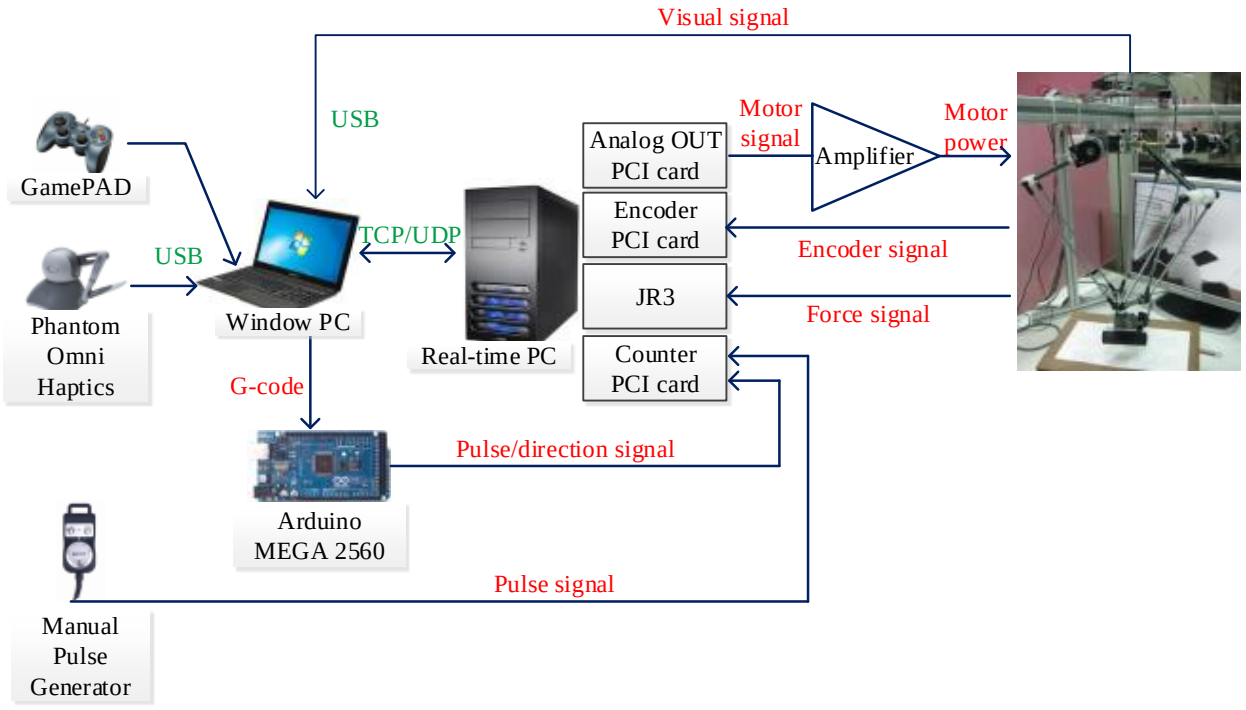

Fig. 1 An Open Architecture hardware platform.

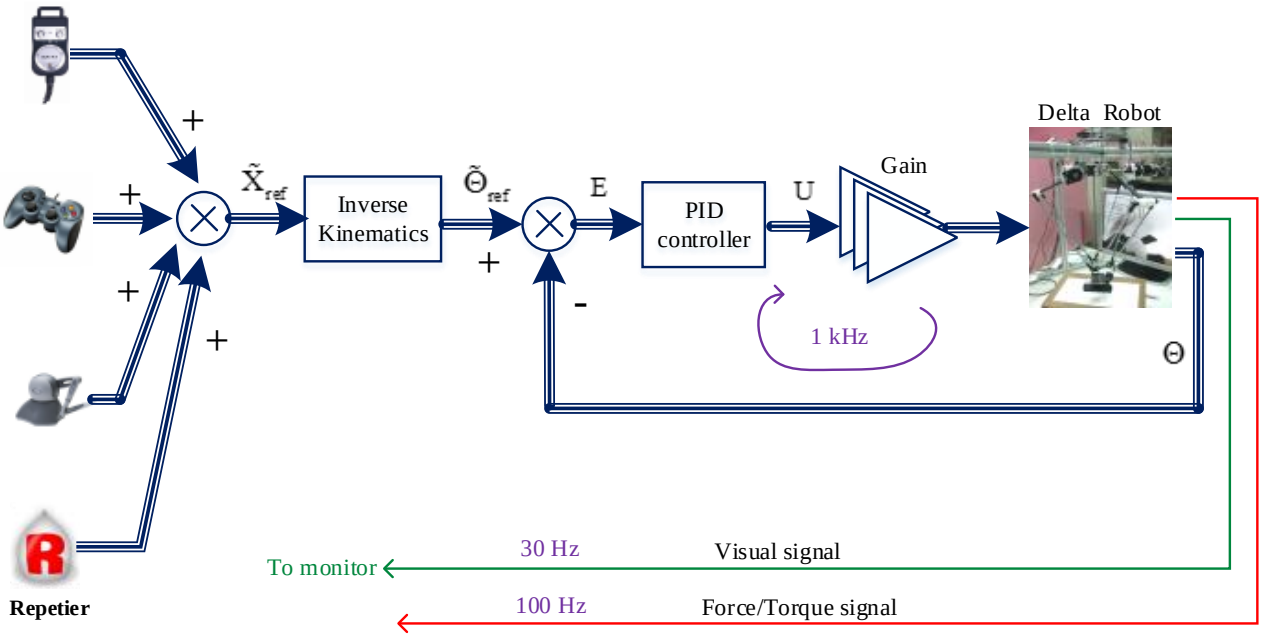

Fig. 2 A graphical control diagram.

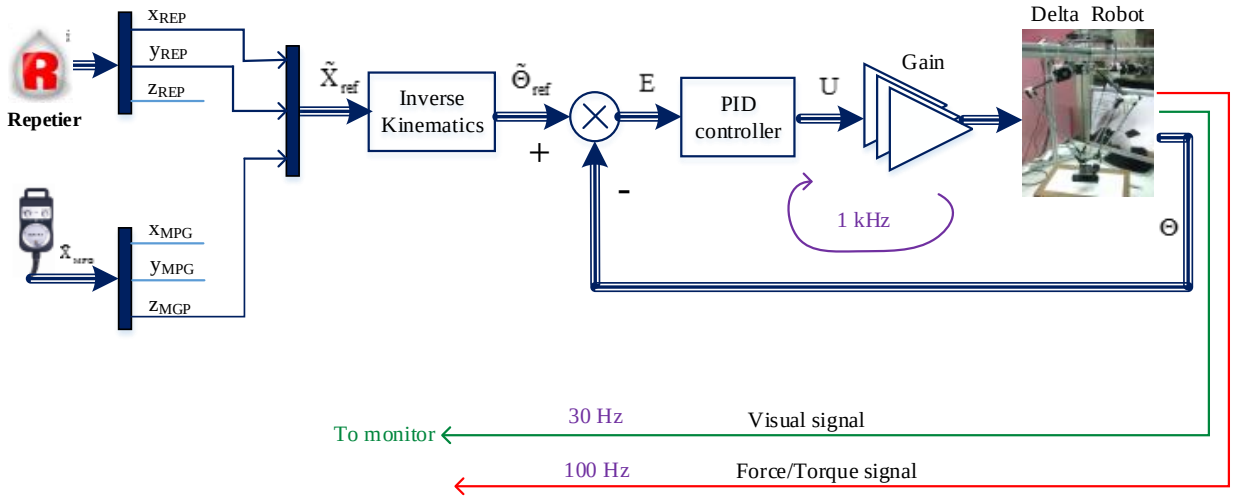

Fig. 3 A control diagram, which is reconfigured such that the *XY* trajectory is driven by G-code while the *Z* trajectory is manually controlled.

The Matlab's Simulink and its xPC toolbox are chosen as a framework to develop control code since it contains all the features and toolboxes that we need to construct the controller. The code, which is quite complicate, is written in graphical language (Simulink) which is easy to reconfigure and modify and bugs are easily detected [12].

Various modules are developed as rigid blocks (shown in Table. 1) especially robot's kinematics and Jacobian and all the I/O cards. The I/O cards are installed in the real-time computer, and can be used in the top level code which is reconfigurable. The blocks can be integrated with the powerful Matlab's Simulink blocks.

Table. 1 The developed Simulink blocks.

I/O	Functions
Advantech 1784 → $\tilde{X}_{MPG}$, $\tilde{X}_{Arduino}$	Send the reference Cartesian position from MPG and Repetier.
Adlink 8133 → $\tilde{\Theta}_{encoder}$	Send the joint angles read by encoder card.
$\tilde{X}$ → Adlink 6308	Command the joint's torques to all the motor.
JR3 → $\tilde{F}$, $\tilde{T}$	Send the force and torque from the JR3 memory.
Mathematics block	**Functions**
$\tilde{\Theta}$ → Forward Kinematics → $\tilde{X}$	Convert the joint to the Cartesian position.
$\tilde{X}$ → Inverse Kinematics → $\tilde{\Theta}$	Convert the Cartesian back to the joint position. (The Cartesian position must be within workspace)
$\tilde{\Omega}$ → Jacobian → $\tilde{V}$	Convert joint velocity to Cartesian velocity.

Open source codes from Matlab and C communities can also be, either directly or with slightly modification, integrated to the top level code. This is a very comprehensive technique to develop a real-time advanced code for our controller.

The controller is also designed as a universal controller for various types of robots. The robot personal file, a file that contains the specific parameter of the robot, can be stored in a flash drive. The general and specific codes are separated so that the controller can effectively control various robots with similar control diagram.

3. Co-processors Controller

The developed controller utilizes two powerful PC processors, one to handle high speed real-time processing at constant 1 kHz and also interface with high speed I/O cards including encoder and analog output cards, and the another is to run Windows based programs, including Matlab, Repetier, Slic3r, and Visual C++ and interface with Windows's devices. The communication between both computers is at about 100 Hz via high speed TCP/UDP. This is a key to handle both real-time and non-real-time codes at the same time and also USB and PCI devices, especially newly one in the market.

4. Interfacing Repetier with Matlab®

One feature of our controller is that it is developed based on software with strong community for developer. In the community, we can share ideas, problems, techniques and even codes. The proposed controller should benefit from the newly proposed techniques and codes from the community. The Matlab® is ideal to our case since it can, (1) handle heavy mathematics through a number of toolboxes, (2) manage codes that run at different servoing rates, (3) solve ODE in runtime, and (4) implement on real-time computer that run at a high servoing rate. However, there is no G-code interpreter in Matlab and its community. The Matlab® does not effectively handle sequential logic and loop, even with its Stateflow®, code that run at varying time, and ASCII text that are all required for G-code programming. In this project, Repetier and Slic3r are chosen to handle G-code and STL file respectively. Both Repetier and Slic3r are very good open source programs that are designed for DIY 3D printer. Roughly, Slic3r is used to convert STL model into the G-code and Repetier is used to generate the command signal at runtime from the resulting G-code.

In the proposed controller, Repetier interfaces with Matlab through PCI counter card installed in the real-time computer. The real-time code in the controller receives the command position from various input simultaneously including Repetier and Manual pulse generator. There is a software selectable switch to manage the position references from various devices.

5. Preliminary test

The proposed reconfigurable open architecture controller is used to control the Delta robot (Fig. 4). The pen and digital camera (Fig. 5) are set up, to draw complex trajectory, in plane, defined in G-code where the height is controlled using manual pulse generator with the visual feedback from the digital camera. The robot is first commanded by manual pulse generator to

the print position. Then, the controller receives the signal from Repetier to draw the G-code trajectory. Roughly, Slic3r converts the CAD model in STL format into the G-code and Repetier processes further to generate the pulse command from G-code. The robot's controller then receives this command and controls the robot to follow the trajectory.

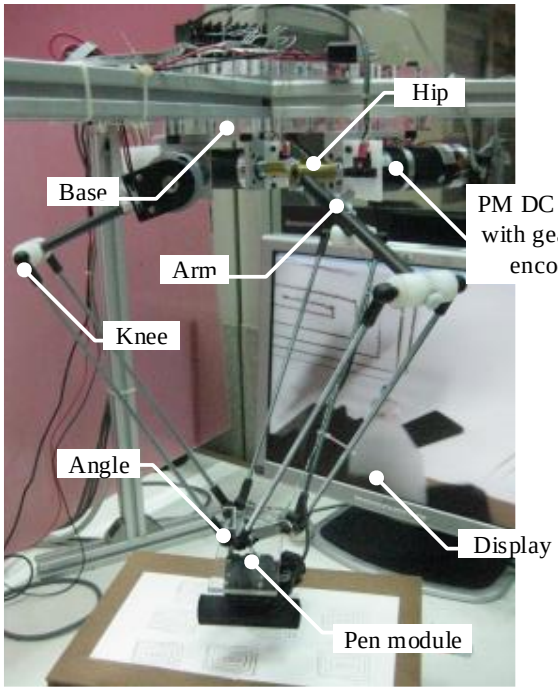

Fig. 4 Our design and built Delta robot.

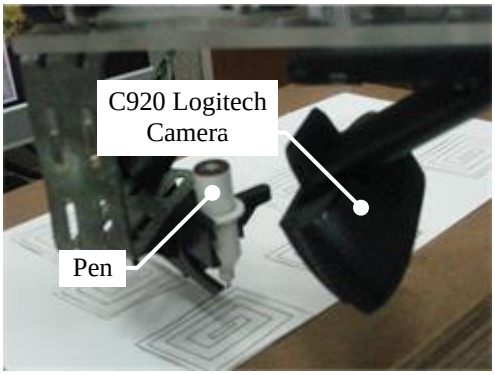

Fig. 5 The Print Head.

The controller is successfully implemented to control the robot as mentioned, the robot go along the trajectory that is generated by CAD model as shown in Fig. 6.

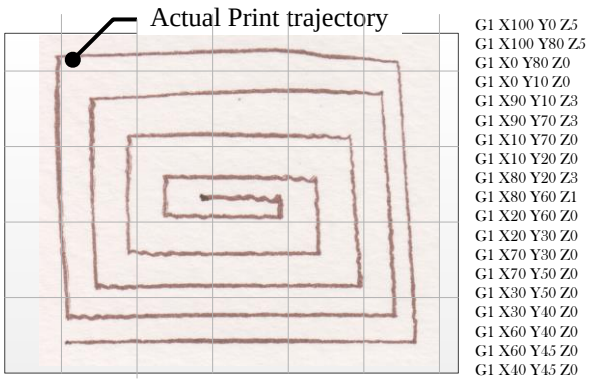

Fig. 6 Printed result commanded by G-code.

However, the actual trajectory is deviated from the G-code command by visual inspection. This is because the robot is not highly rigid. There is no gravity and interaction force compensations and only rough calibration is performed.

6. Conclusion

This preliminary result demonstrates the capability of the controller as it can interface with manual pulse generator and Logitech C920 camera and can control the motion of the Delta robot. Furthermore, the motion can be defined in G-code command that is generated from CAD model. We have also tested the interfacing with Microsoft's Kinect, Sensible's Phantom Omni, and Logitech's GamePAD that are connected to the Windows based computer and successfully use these devices to control the motion of the robot.

7. Ongoing Tasks

The Matlab's SimMechanics model of the Delta robot (Fig. 7) is also develop to solve robot kinematics and dynamics in real-time. This bond graph model has a strong potential to handle kinematic of robot in an effective way. The Block for JR3 PCI card, the force/torque receiver card, is already developed and under testing. The controller is also planned to test with CRS articulated robot and FANUC SCARA robot.

8. Acknowledgement

I would like to thank you Mr.Jaruboot Kananai and Mr.Narit Boonhaijaroen, senior researchers in robotic lab for their valuable advices and the Delta robot that is used to test our controller. The authors also thanks the Ratchadaphiseksomphot Endowment Fund of Chulalongkorn University (RES560530127-AS) for financial support.

9. References

[1] John J. Craig (1989), *Introduction to ROBOTICS: mechanics and control*, 2nd edition, Addison-Wesley Publishing Company, 0-201-09528-9, USA.

[2] A. Blomdell, G. Bolmsjo, T. Brogardh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, J. Wang, Extending an Industrial Robot Controller : Implementation and Applications of a Fast Open Sensor Interface, *IEEE Robotics & Automation Magazine* 2005, pp. 85-94.

[3] D. C. Santini, W. F.Lages, An Open Control System for Manipulator Robots, paper presented in *the 20th International Congress of Mechanical Engineering*, 2009.

[4] A. Oliveira, E. De Pieri and U. Moreno, An Open-architecture Robot Controller applied to Interaction Tasks, Advances in Robot Manipulators, Ernest Hall (Ed.), ISBN: 978-953-307-070-4.

[5] W. E. Ford, What is an Open Architecture Robot Controller?, paper presented in the *IEEE International Symposium on Intelligent Control*, 1994.

[6] K. S. Hong, J. G. Kim, C.-D. Huh, K. H. Choi, and S. Lee, A PC-Based Open Robot Control System: PC-ORC, paper presented in the *IEEE International Symposium on Industrial Electronics (ISIE)*, 2001.

[7] D. C. Santini, W. F.Lages, An Architecture for Robot Control Based on the OROCOS Framework, paper presented in *Robocontrol*, 2010.

[8] Repetier, the software driving your 3D printer, URL: http://www.repetier.com/

[9] Slic3er, G-code generator for 3D printers, URL: http://slic3r.org/

[10] Wikipedia the free encyclopedia. *G-code*, URL: http://en.wikipedia.org/wiki/G-code.

[11] P. Liandong, H. Xinhan, Implementation of a PC-based Robot Controller with Open Architecture, paper presented in the *IEEE International Conference on Robotics and Biomimetics*, 2004.

[12] K. K. Tan, K. Z. Tang, H. F. Dou, S. N. Huang, Development of an Integrated and Open-Architecture Precision Motion Control System, *Control Engineering Practice*, volume 10, issue 7, July 2002, pp. 757–772

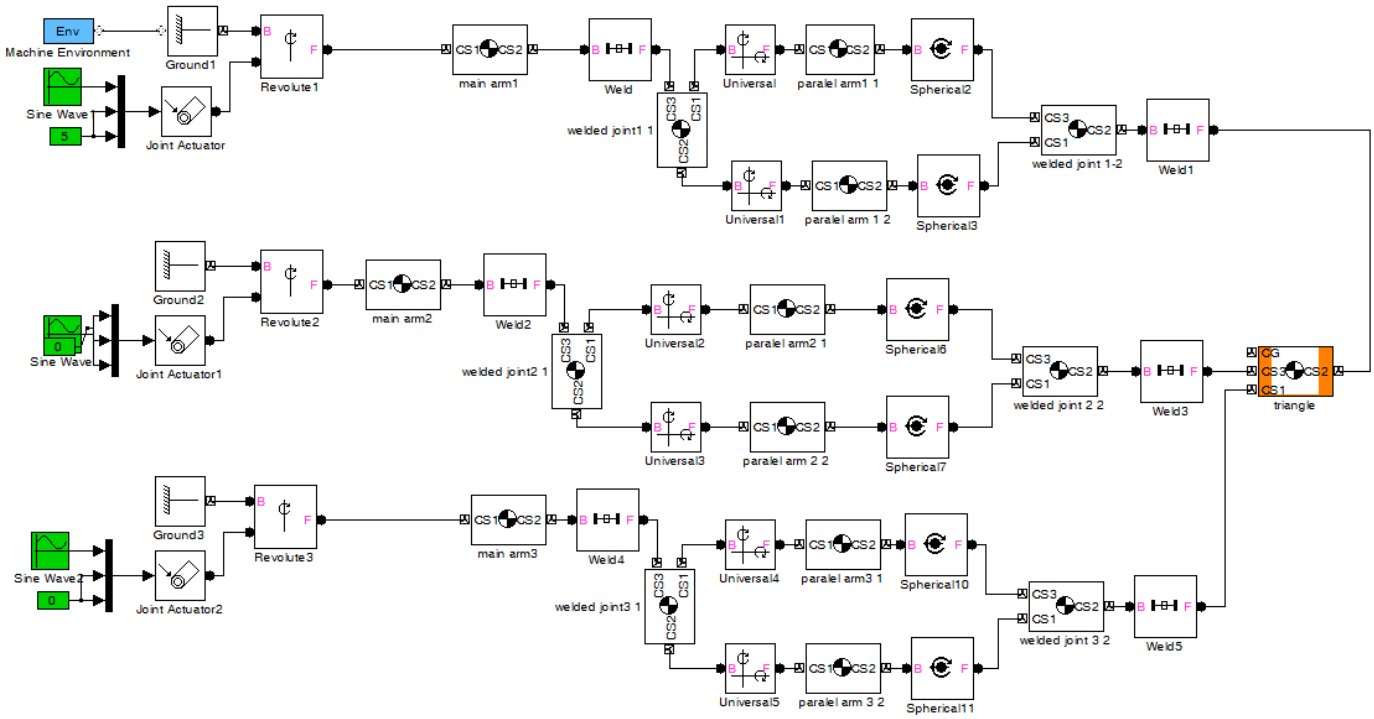

Fig. 7 The SimMechanics model of the Delta Robot.