MSA UNIVERSITY ENGINEERING JOURNAL

Upgrading Gryphon PUMA Robot Arm Using ROS-MoveIt

Ahmed E. Newir[1, a], **Youssef A. Abdelfattah**[1, b], **Mostafa M. Rashed**[1, c]
1 Mechatronics department, Faculty of Engineering, October 6 University, October, Egypt
E-mail: [a]ahmed_newir.eng@o6u.edu.eg, [b]youssefabdelmajeed77@gmail.com, [c]mostafarashed11001@gmail.com

Abstract

Industrial and old lab robotic equipment can be upgraded and reused in very sophisticated applications such as robot arms if provided with an efficient modular program or framework such as ROS (Robotics Operating system). In this paper, we replace the old control system of the 5-DOF robot arm (complete Gryphon Package 35-002-c) with the help of ROS MoveIt Motion Planning Framework package to control motion planning and generate joints angles, URDF (Unified Robot Description format) used to create the robot model for ROS. The microcontroller is used to move the five stepping motors to move the end effector in the desired pose: position and orientation using ROS and MoveIt capability. Thus, the robot should be upgradeable, and anyone who has ROS basic skills can improve the arm functionality to use it in any desired application.

Keywords: ROS1 (Robotics Operating system), MoveIt, Robotic manipulator, gryphon robot, 5 DOF industrial robot arm.

ENGINEERING JOURNAL Volume # Issue #
Received Date Month Year
Accepted Date Month Year
Published Date Month Year
Online at https://engj.org/
DOI:10.4186/ej.20xx.xx.x.xx

1. Introduction

The gryphon PUMA robot arm is a part of the robots family designed by the WALLI group, which are products of Italtec Company; the arm consists of five stepping motors controlled by four processors a CNC processor as a master and three configured as slaves controlled by WALLI software [1]. In addition, this arm was used in both educational and industrial applications such as pick and place operations.

Singularity for 5-DOF gryphon robot calculated by [2] to use it in industrial applications. [3]replace the old control with Simulink® environment of MATLAB® and use Tele-Visual tool kit, also used a camera to make image processing to control the gryphon arm robot and found a visible delay in controlling robot motion in the next research [1].

Robot operating system [4] and MoveIt [5] are open sources used to operate and manipulate different robots. Many robots using ROS as [6] present their experiences and results while building an autonomous robotic assistant using the PR2 platform and ROS. Also [7] uses ROS to make intuitive programming of dual-arm robots and concludes that reduction in the programming complexity, requires fewer steps. Easily interface of the 3 (DOF) arm robot designed and operated using Robot Operating System (ROS) and MoveIt library.

In [8] show the need for help from experts in all the areas that make up the interdisciplinary field of robotics, from low-level control to high-level reasoning, and everything in between. With the goal of enlisting those experts as contributors, we made three key design decisions in ROS.

The current configuration of the robot arm cannot be reused in complex applications especially because the arm program shown in fig. 1 is not open source and its control unit is not compatible with recent computers motherboards, then the only left solution is to build a new control system, and this is relatively hard. Fortunately, ROS or Robotics Operating System has the tools that can aid with this problem, and one of these tools is the MoveIt package which is a collection of programs that work together to interact with a robot model. The Unified Robot Description Format (URDF) model is used to do motion planning and object manipulation and Inverse kinematics by solving for the joint position for a given position and orientation on this model which can be used by a controller to achieve Hardware-in-the-loop simulation with the physical hardware.

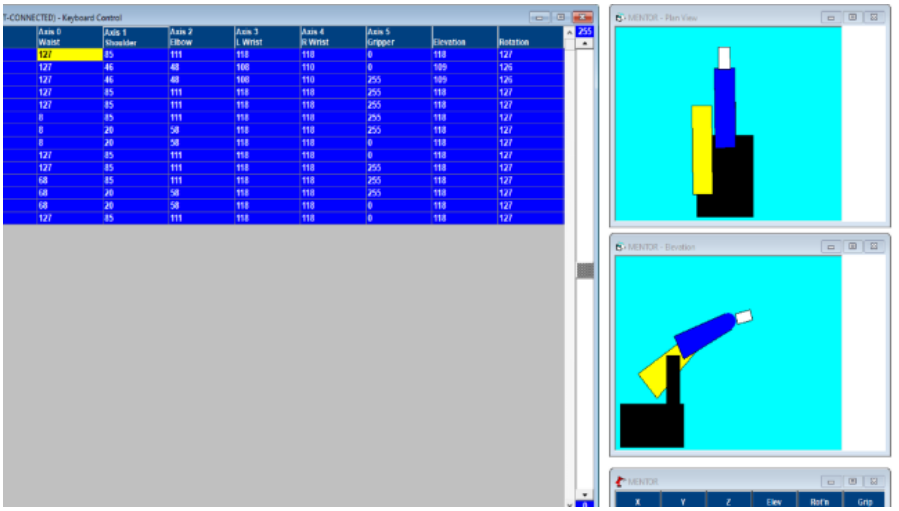

Fig. 1. WALLI4 Program Interface.

The proposed steps to improve the current configuration:

1) Create a 3D model of the robot arm.
2) Visualization of the robot arm movement with a better Graphical User Interface.
3) Create or interface with a solver to solve the arm kinematic model.
4) Movement controller implementation.

2. Arm 3D Model & Visualization

To improvable the arm system, the system should be built on a universal robotic framework. We use Robotics Operating system (ROS) which is a free open-source meta-OS that runs on Linux distributions, and to have a visualization of the robot model in that framework Unified Robot Description Format (URDF) file should create describing each joint metadata in Extensible Markup Language (XML).

One of the easiest methods to create a robot description is to create a 3D model and export it using one of the provided plugins. The 3D model designed for the gryphon arm robot shown in Fig.2

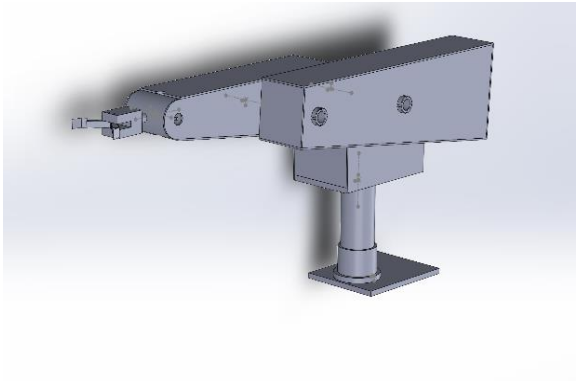

Fig. 2.a. Gryphon Robot Arm 3D Model.

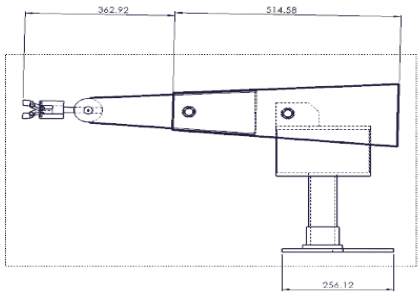

Fig. 2.b. Gryphon Robot Arm 2D model.

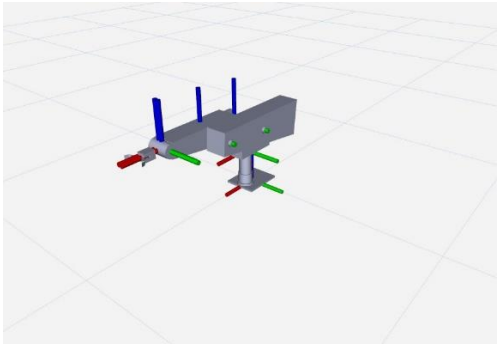

Fig. 3 a. ROS- RViz Gryphon Arm URDF Visualization.

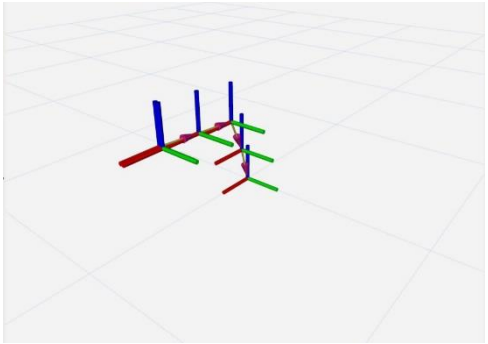

Fig. 3 b. ROS- RViz Gryphon Arm TF Visualization.

Fig. 3 represents gryphon arm robot Unified Robot Description Format (URDF) using ROS RViz to handle the transformation (translation-rotation) of each joint of the robot arm in the background by using the TF package.

3. ROS-Moveit Setup

Fig. 4. ROS-MoveIt Setup process

Fig. 4 shows the setup of a robotic arm URDF model in ROS MoveIt first specify all link of the robot that always collide which decreases the time needed to solve for robot angles [9]. Then a fixed joint must be specified which defines the robot body's relation with the world this joint is called a Virtual joint. Then the robot end effector must be specified, and it is the link before the actual robot end effector link. Then for each planning group defined a ROS controller should be associated with it and ROS Controller can be (position, velocity, effort) controller [10].

4. Implementation Of the Controller

4.1. controller programs

Fig. 5 shows robot nodes (programs) used in controlling the robot arm, first Joint_states GUI sends angle data in rad to a message called joint states, which transfers it to two programs. Robot_state_publisher, which uses that data to visualize arm movement, and to

Gryphon_states_handle program, which converts the angles to steps, used by the gryphon controller program to move the actual arm using stepper motors.

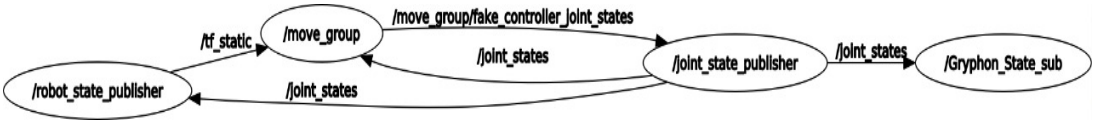

Fig. 5. The ROS graph of Gryphon robot nodes.

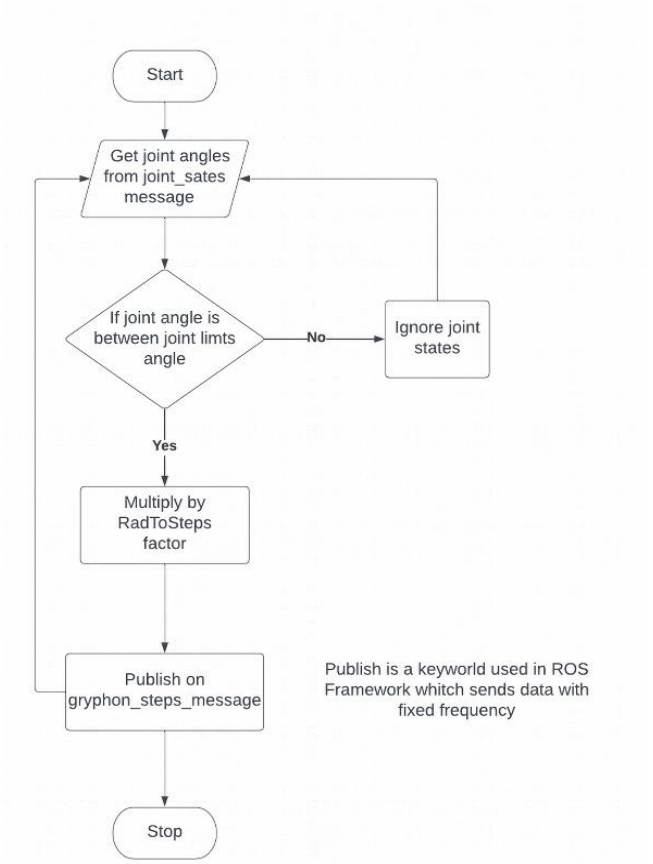

Fig. 6. Gryphon states handle program (node) flow chart.

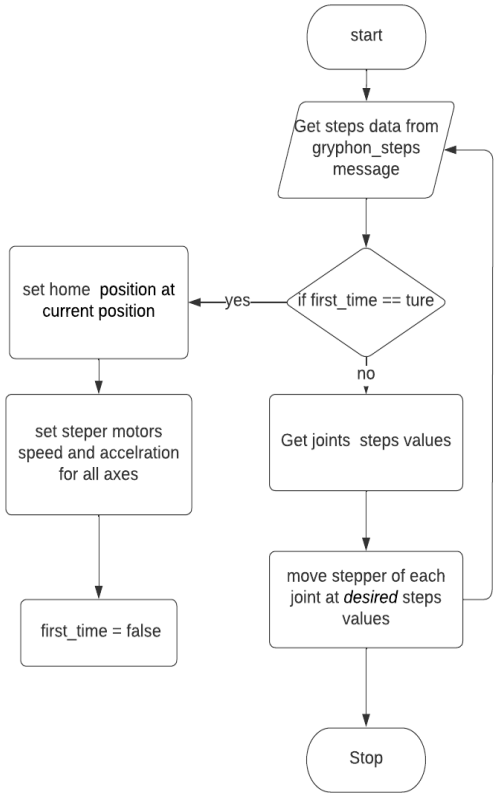

Fig. 7. Gryphon controller program (node) flowchart

4.2. Wiring Schematic

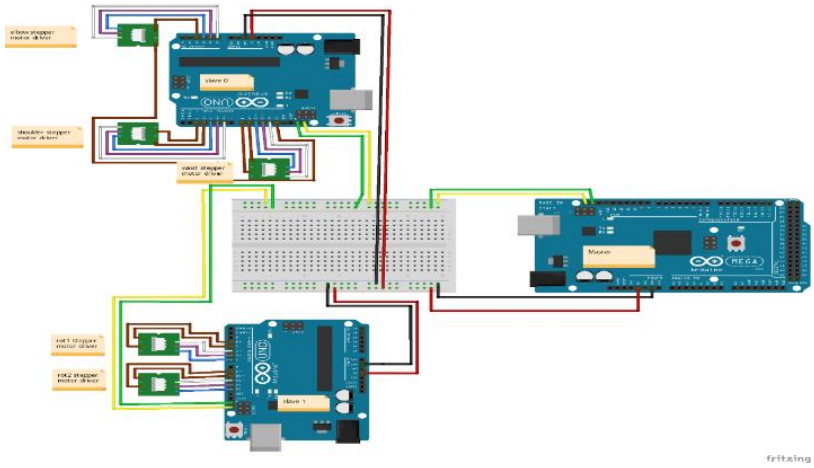

Fig. 8. Modified Gryphon wiring diagram

The update on the gryphon robot arm control unit consists of three AVR microcontrollers. One Arduino mega 2560 as a master which is used to read serial data from the pc host machine and command the two Arduino UNO with the needed number of steps for each joint stepper motor. Each Arduino UNO handles 3 joints of the robot arm to avoid any movement delay.

4.3. Arm Movement Calculations

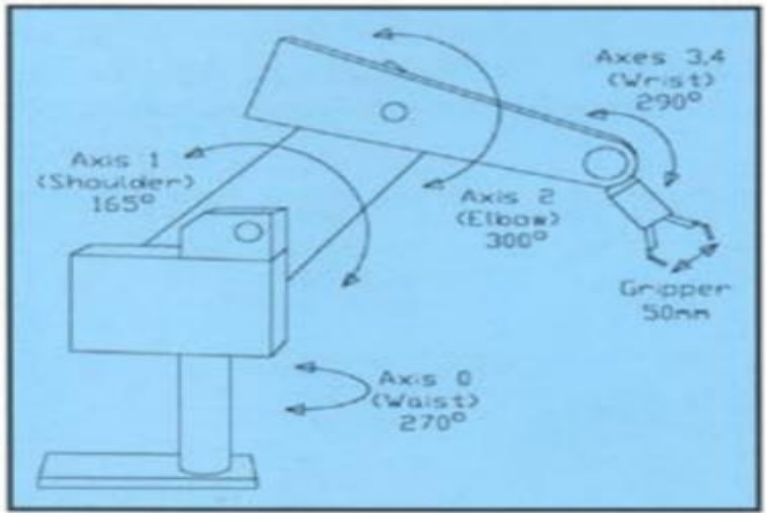

Fig. 9. Gryphon Angular Movement Diagram

The encoder attached to each link and each link is different in motor size and gear ratio resulting in different encoder pulses per revolution as in table 2.

In addition, to calculate how many steps for each motor per one Rad drive the motor arbitrarily with a known number of steps then measure the corresponding angle traveled then work out the steps per rad table 3 shows the measured angle for the first three axes when moved 12,000 steps.

Table 1. Axes angular movement

NO	Name	Angular movement (Degree)	Angular movement (Radian)
0	Waist	270	4.71299
1	Shoulder	165	2.87979
2	Elbow	300	5.323599
3	Left wrist	290	5.06145
4	Right wrist	290	5.06145

Table 2. Axes pulses per revolution

NO	Axis	Encoder pulses per revolution
0	Waist	4600
1	Shoulder	4000
2	Elbow	5000

Table 3. Measured angles for the first 3 axes when moved 12,000 steps

NO	Axis	Angle (Radian)
0	Waist	2.3041
1	Shoulder	1.8240
2	Elbow	3.1200

To work out the rad per step factor is needed to know the number of steps in one rad and the used formula is:

$$\frac{number\ of\ steps}{angle\ travllerd\ (rad)} \quad (1)$$

Table 4. Number of steps per one rad for the first three axes

NO	Axis	Steps Per Rad
0	Waist	5208.10729
1	Shoulder	6578.94737
2	Elbow	3846.15385

5. Conclusion

This paper presents a simple solution to upgrade the firmware of any robot arm using ROS- MoveIt to solve the inverse kinematics of the arm using planning groups. In addition, how to implement a proper controller to convert ROS controller output angles to steps controller the stepper motors thus moving the arm.

Though we believe that we have made good steps towards upgrading the gryphon robot to ROS with MoveIt error calculations have not been identified yet, and many issues remain, such as adding some sensors to calculate the difference between the actual and ROS MoveIt trajectory with many trials. In addition, trying to check different algorithms of PID motion controller although the results are outwardly good.

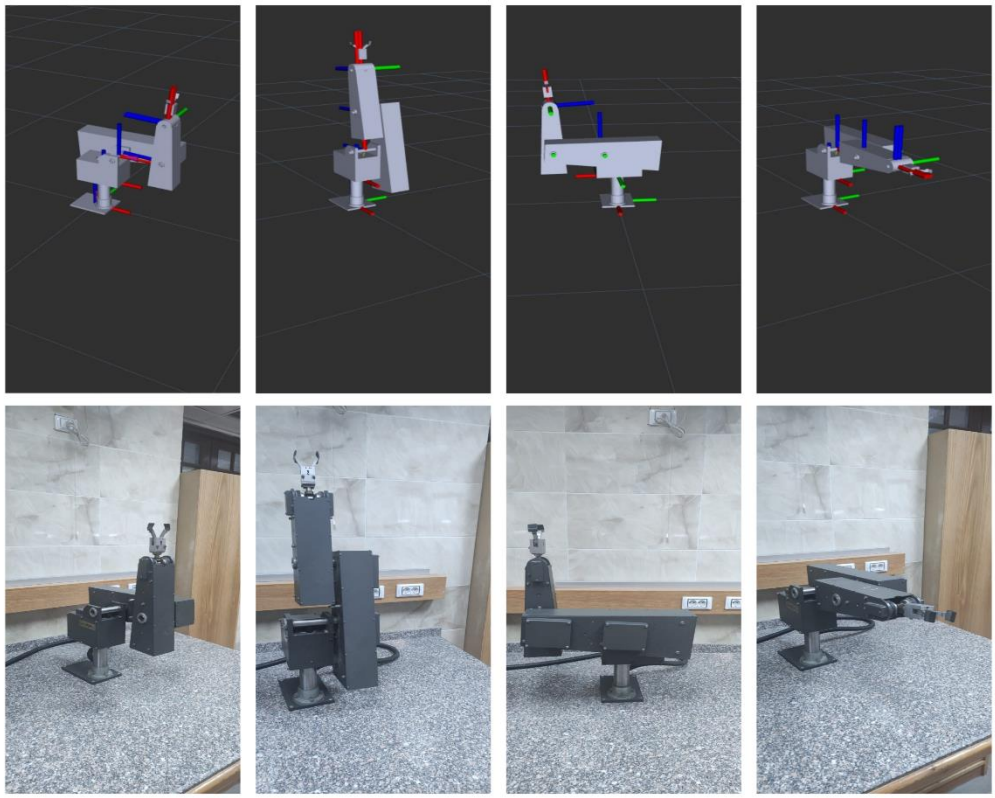

Fig. 10. Gryphon Motion with the updated controller

When designing the controller there were no disturbances affecting the arm motion (external forces) there was the whole idea for demonstrating the purposes and because of that and stepper Motors's repeatability, an open loop controller was implemented and achieved fairly good position control results as shown in Fig.10.

References

[1] Hassanzadeh, I. a. (2009). Tele-Visual Servoing of Robotic Mamipulators; Design, Implementation and Technical Issues. *Journal of Applied Sciences 9.2*, 278-286.

[2] Cheng, F. T. (1995). Analysis and resolution of singularities for a 5-DOF GRYPHON manipulator. *In 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century* (pp. 4416-4421). IEEE.

[3] Hassanzadeh, I. a. (March 2007). Open Architecture of Gryphon Robot for Visual Servo and Teleoperation Tasks. *Proceeding of the 4th t International Symposium on Mechatronics and its Applications (ISMA07)*, (pp. 26-29). Sharjah, UAE.

[4] Quigley, M. K. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source software Vol. 3, No. 3.2*, p. 5.

[5] Chitta, S. I. (2012). Moveit![ros topics]. *IEEE Robotics & Automation Magazine 19, no. 1*, 18-19.

[6] Bohren, J. R.-E. (2011). Towards autonomous robotic butlers: Lessons learned with the PR2. *In 2011 IEEE International Conference on Robotics and Automation* (pp. 5568-5575). IEEE.

[7] Makris, S. P. (2014). Intuitive dual arm robot programming for assembly operations. *CIRP Annals, 63(1)*, 13-16.

[8] Cousins, S. B. (2010). Sharing software with ros [ros topics]. *IEEE Robotics & Automation Magazine 17, no. 2*, 12-14.

[9] Perdomo, E. F. (2015). *Learning ROS for robotics programming: your one-stop guide to the Robot Operating System.* Packt Publishing.

[10] Joseph, L. a. (2018). *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System.* Packt Publishing Ltd.