New Orleans JAZZED about Engineering Education

Paper ID #14789

Low-Cost Robot Arms for the Robotic Operating System (ROS) and MoveIt

Dr. Asad Yousuf, Savannah State University
Mr. William Lehman, Bill's Robotic Solutions

William Lehman is President of Bill's Robotic Solutions which he started in July of 2013. He has had over twenty years of experience in software and hardware development. He has worked on numerous projects in digital communication systems, robotics, and aerospace applications. Mr. Lehman received his Bachelor of Science degree in Electrical Engineering in 1979 from Catholic University of America.

Dr. Mohamad A. Mustafa, Savannah State University

Mohamad Mustafa is a Professor of Civil Engineering Technology and the Chair of the Engineering Technology Department at Savannah State University (SSU). He has six years of industrial experience prior to teaching at SSU. He received his BS, MS, and PhD in Civil Engineering from Wayne State University, Detroit, Michigan.

Dr. Mir M. Hayder, Savannah State University

Dr. Hayder is an Assistant Professor in the Department of Engineering Technology at Savannah State University, GA. He received PhD in Mechanical Engineering from McGill University, Canada. His research interest lies in the areas of engineering education studies, robotics, fluid-structure interaction, flow-induced vibrations, syngas and blended fuel combustion, nanofluids, concentrating solar power technologies, and flow and structural simulations.

LOW COST ROBOT ARMS FOR THE ROBOTIC OPERATING SYSTEM (ROS) AND MOVEIT

It is not uncommon for students in high school and college to design and build low cost robot arms. This paper summarizes the results of an undergraduate assignment to design and build a low cost robot arm, as well as a robot arm controller. The robot arm controller uses accelerometers to control the motion of the robot arm. The robot arm controller can also be used to record and playback a sequence of motions for the arm. The robot arm controller was Arduino Uno Micro-controller based to keep costs down. A serial interface was also implemented for the arm controller so the arm could be controlled from a PC. The students had a mentor from industry to guide them in the design of their robot arm and controller. The mentor also evaluated the robot arm and similar designs for use with the Robotic Operating System (ROS) and Moveit software, for possible use of Moveit on future student projects.

ROS and Moveit bring interesting functions for control of robot arms. The Open Motion Planning Library (OMPL) is used by the Moveit, providing a variety of motion planning algorithms to control the students arm. A 3D Camera can be directly used by Moveit to provide obstacle avoidance functions for the robot arm. The results of the evaluation of Moveit were shown to the students in a video as well as the other results of the evaluation giving them insight into how an embedded sub-system they developed can interact as part of a complex system.

Introduction

The field of robotics by nature is an interdisciplinary endeavor. Robots have been applied in numerous applications from the bottom of the ocean to Mars and beyond. Our Electronic Engineering Technology (EET) students have great interest in the growing field of robotics. Industrial robots arms have been used over fifty five years in industry and have been enormously successful. Robots in industrial setting have traditionally depended upon very structured environments where materials are carefully positioned. Recently applications in industrial robots have been applied to less structured environments due to advances in technology [1].

To introduce industrial robotics we decided to use the Open Source Robotics Operating System (ROS) and Moveit software [2]. In previous work we developed labs using ROS and Moveit to teach basic Kinematic concepts in robotics [3]. The previous work only used simulated robot arms to teach concepts in robotics. The obvious next step for us was to add a real robot arm to the lab configuration.

We have worked with well over a hundred students at the high school level and a small number at the college level successfully building robot arms similar to the one shown in this paper. The robot arm used RC servo motors to drive the robot arm joints. The robot arm was controlled in all cases by switches or Potentiometers (POTS) on the Micro-controller. The Micro-controller was not connected to the PC. We decided to use the same robot arm design for use with ROS and Moveit.

Because of the complexity and high learning curve in using the ROS software system we used a mentor from industry to help students design and build the robot arm and program the micro-controller. The students programmed the Arduino Mircro-controller to control the robot arm movements with Potentiometers (POTS) or Gyro. The mentor wrote code for the Arduino Micro-controller to work with ROS and Moveit. The mentor also evaluated the Robot Arm performance with ROS and Moveit.

We have not yet had students use the robot arm with ROS and Moveit, since we needed to insure this was a viable option. This paper details the lab design using the robot arm and the results of the evaluation of the robot arm for use with ROS and Moveit. To understand the lab design it is necessary to understand the Lab Hardware, Arm Mechanical Design, ROS Framework and Architecture of Software. Finally we discussed the tests to evaluate the robot arm and results.

Lab Hardware

Figure 1 shows the hardware configuration of Robotics Lab. The Ardunio Uno Micro-controller is the robot controller. Ubuntu Linux is the operating system for the PC. ROS and Moveit runs under Ubuntu on the PC. A Microsoft Kinect camera is used to detect obstacles blocking arm motion. The Gyro (Parallax 3-Axis L3G4200) and Servo Motors were connected to the Ardunio Uno board [4]. The Gyro was offered to the students for extra credit, as a alternative to using POTS to control the robot arm. The Gyro was not used by the mentor when evaluating ROS and Moveit.

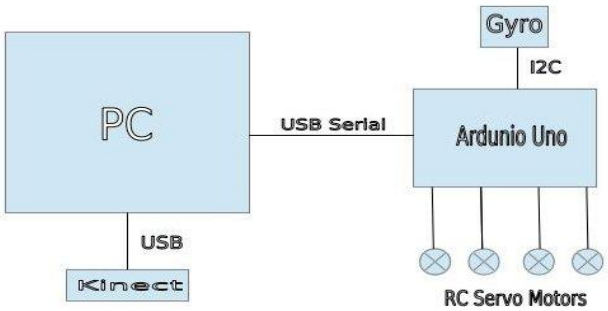

Figure 1 Hardware configuration for robot arm

Arm Mechanical Design

The robot arm was a simple under-actuated 3R robot. The robot model for the robot arm is shown in Fig 2. The robot model contains the relationship of link frames with respect to the base link. There are only three revolute joints since a hook; the last link in the robot model is used to pick up the paper forms.

The robot model display depicted in Fig 2 was generated by the RVIZ robot simulator. Note this view of the model only shows the details of the kinematic chain [5].

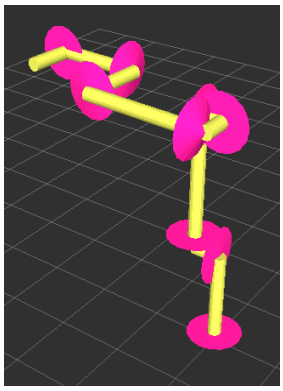

Figure 2 Robot model

The robot model as displayed in Figure 2 was written in the Unified Robot Description Format (URDF). URDF models many features of robots. We used URDF to model the following features of the Robot Arm:

a) Kinematic Tree
b) Visual Display
c) Collision Model

It is essential that the real robot matches the Kinematic Tree modeled. Any error in modeling the Kinematic Tree results in systematic error between motions and positions of the real robot verses the simulated robot in RVIZ. The Visual Display and Collision Model were both approximations of the robot arm. The Visual Display of the robot arm did not include the RC Servo Motor case and did not include the rectangular base of the robot. The Collision Model was a rough approximation of the robot arm body that was sufficient for the experiments in obstacle avoidance we performed.

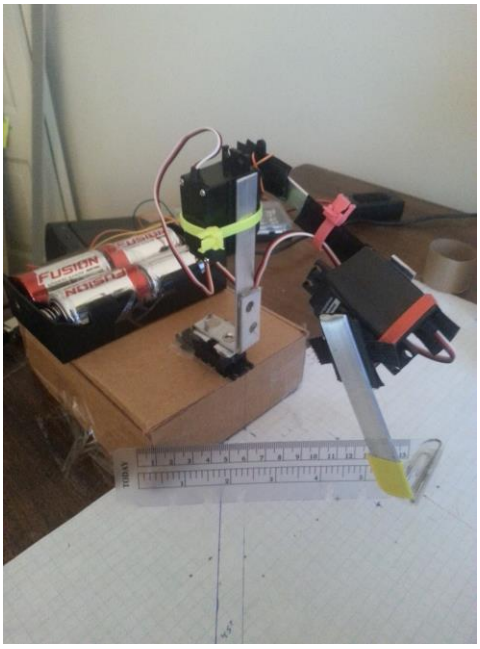

Figure 3 Robot arm

Figure 3 shows a photo of the robot arm. The upright link is approximately 10 cm, with the second link 14 cm and the third link 8 cm in length. Given the forms picked up by the arm are made of paper and the small form factor of the arm, the RC Servo Motors provided more than enough torque.

It was made out of ½ by ¼ inch aluminum strips cut into short pieces and held together with using a hot glue, wire ties and Velcro. We have also made similar robot arms using cardboard as the main material. The three joints were moved by three RC Servo Motors.

This simple design was chosen as a proof of concept. The software as discussed in the next two sections is flexible and supports robot designs with up to 12 joints.

Overview ROS Framework

In order to understand how the ROS System works, we need to have a high level understanding of the ROS Framework. ROS is based upon distributed computing with the "node" as the basic computational entity. Each ROS system is composed of one or more nodes.

Figure 4 is a diagram that represents some of the major features of a ROS node.

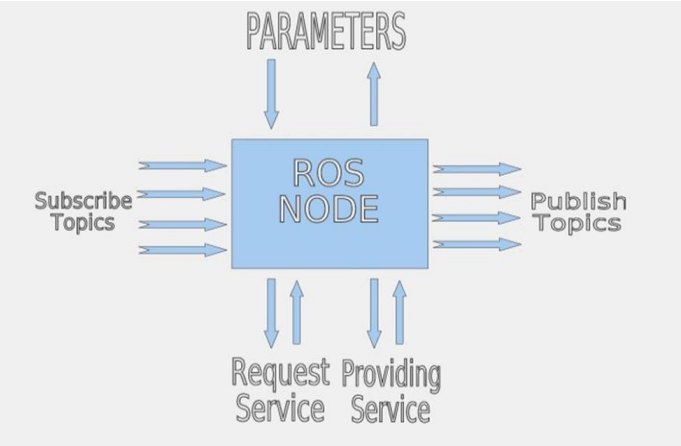

Figure 4 ROS node

Nodes are useful because they communicate with other nodes. Figure 4 shows the main way nodes communicate with one another:

- Parameters
- Services
- Topics

In addition to nodes every ROS system has a single Parameter Server that stores and retrieves Parameters (variables) used by nodes. Unlike Parameters in a program or Operating System these Parameters can reside on different computers. As with all communication to and from a node the information is sent over the Internet.

Services provide node with a client/server mechanism to communicate with another node. We did not use Parameters or Services in the software we wrote for this project. We only mention them to give some background for the reader. Topics were used in nodes we wrote for this project.

Topics provide a communication mechanism to send a message from one node to many other nodes. The node that sends the message is called the Publisher. There is only one Publisher node for a given topic. The nodes that receive the message are Subscribers to the Topic [6].

The downside of Topics is the Subscribers are not guaranteed no messages were lost. In spite of this we found the communication link acceptable for sending trajectories to our robot controller. The upside of using topics is that we were able to use the Fake Robot Controller. We also were able to take advantage of the Robot State Publisher. The Robot State Publisher with GUI allows the robot to be controlled with sliders in a window on the PC screen.

Architecture of Software

In order to understand how a ROS System works we need to know the precise configuration of ROS nodes and the communication mechanisms they use to talk with one another. We used three different ROS Systems to control the robot arm lab. We will discuss each of the systems and how the nodes are configured in each system.

In the first system, the arm is controlled manually using the Joint State Publisher. The Joint State Publisher displays a GUI screen with sliders for each joint of the robot. Figure 6 shows the Joint State Publisher node publishes a Joint States topic which is used by the RVIZ simulator and Brazen Joints node. The Brazen Joints node converts the Joint State message to a Brazenbot message. The ROS serial node receives the Brazenbot message and passes it over the USB serial connection to the robot controller node [7].

In the second system, Moveit is brought up in demo mode and the Fake Robot Controller generates Joint States. The Joint States are processed the same way as in the first system, after being generated by Moveit.

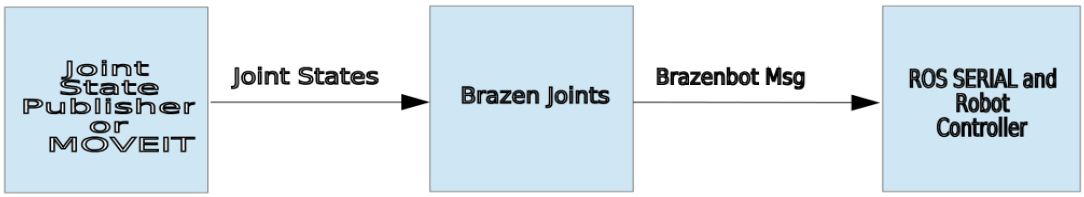

Figure 6 Software configurations of ROS nodes

We also tested a third system, using a Modified Fake Robot Controller. In the two systems previously discussed we did not modify Moveit in any way. We Modified the Fake Robot Controller to generate our Brazenbot Message. We did not need the Brazen Joints node we developed in this case, since the Brazenbot Message is passed directly from the Modified Fake Robot Controller directly to the ROS Serial node. Figure 7 shows the configuration of nodes used in the third ROS System [8].

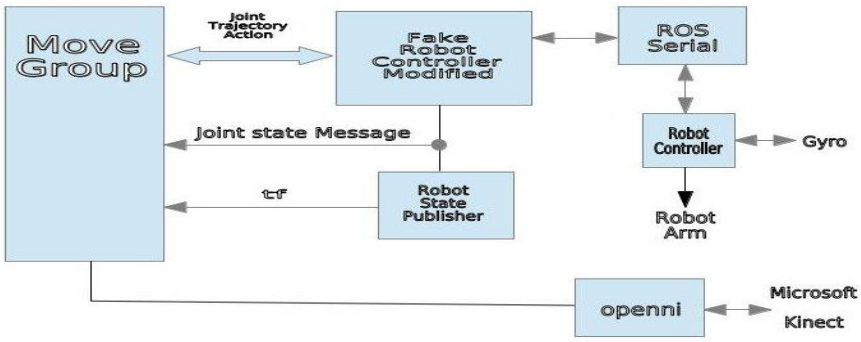

Figure 7 ROS system Modified Fake Robot Controller

The third ROS System with the Modified Fake Robot Controller works like the first system from a user's perspective. We modified the Fake Robot Controller since it is a good place to process feedback from the robot controller running on the Ardunio Uno Micro-controller. Currently there is no feedback from the Micro-controller, but we are considering adding sensors to the robot arm in the future.

ROS and Moveit Evaluation Tests

There were three main areas of testing performed with the robot arm:
a) Accuracy tests
b) Moveit path planning tests
c) Obstacle avoidance tests

Paper in the form of tubes/cubes at various heights marked the physical location in three dimensions. Fine tuning of the robot model was made at this point under manual control. The difference between the simulated and physical locations was recorded. Table 1 summarizes the results of these accuracy tests [9].

	Delta X	**Delta Y**	**Delta Z**
Average Error in Position	1.4 cm	0.6 cm	0.2 cm
Maximum Error in Position	2.1 cm	1.7 cm	0.5 cm
Minimum Error in Position	0.8 cm	0.2 cm	0.0 cm

Table 1 Result of Accuracy Tests

The moveit path planning tests were performed for a set of known test locations. The path planning tests were repeated many times to confirm the robot arm would accurately reposition to the exact same location. The tests points were repeated 20 or more times each and showed the robot would repeat the same position to within 1-4mm.

Moveit has the capability to deal with unstructured environments such as pick and place where obstacles can unexpectedly block the robots movements. Moveit uses 3D camera images to find obstructions and uses path planning to circumvent the obstructions. The openni package is used to communicate visual data from the 3D Microsoft Kinect camera to the Move Group. Move Group processes the 3D point cloud message from openni and builds an octomap of obstacles that surround the robot. An octomap displays obstacles as cubes on the screen. The sizes of the cells that compose the octomap are user selectable [10].

The obstacle avoidance tests repeated a path planning test with an obstacle placed between the start and end goal of the path planning test points.

Dealing with the 3D camera was non-trivial. There are trade-offs where the camera is located in relation to the robots position. We located the camera 0.7m away from the robot arm. The response of the system to the camera's detection of an obstacle was only useful for static obstacles placed in the way of the robot. Response time to detecting an obstacle was in the range of a minute for a 5 mm block size in the octomap.

When the octomap block size was set to 1 mm, the processing time to display a octomap of obstacles took 6 to 7 minutes. We should note we were using one PC to run all the Moveit and camera software.

A 1mm block size would be best for obstacle avoidance due to the small objects and small work area of the robot arm.

Conclusion

We envision many ways other instructors could use a robot arm lab similar to the one we have developed. The software we wrote will be available on http://www.brazenbot.com. The documentation including movies of the robot arm in action will also be available at that site.

We felt we needed to test the robot arm lab hardware to insure it works, before we started spending time and effort in writing curriculum for it. We are currently considering writing curriculum to teach a number of concepts needed by EET students through the robotics arm lab.

References

[1] Corke, Peter (2013) "Robotics, Vision and Control" (pp. 1-6). Springer-Verlag Berlin Heidelberg

[2] "Why teach robotics using ROS". Retrieved January 30, 2016 from https://www.researchgate.net/publication/274267022_Why_teach_robotics_using_ROS

[3] Asad Yousuf, Savannah State University; William Lehman, Bill's Robotic Solutions; Mir Hayder, Savannah State University; Mohamad Mustafa, Savannah State University, "Introducing Kinematics into Robotic Operating Systems", pg. 39-47, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION | V7, N2, FALL/WINTER 2015

[4] Gyroscope Module 3-Axis L3G4200D (#27911) (n.d.). from https://www.parallax.com/sites/default/files/downloads/27911-Gyroscope-3-Axis-L3G4200D-Guide-v1.1.pdf.

[5] rviz (n.d.). Retrieved January 30, 2016, from http://wiki.ros.org/rviz

[6] ROS Computation Graph Level (n.d.). Retrieved January 30, 2016, from http://wiki.ros.org/ROS/Concepts

[7] joint_state_publisher (n.d.). Retrieved January 30, 2016, from http://wiki.ros.org/joint_state_publisher

[8] System Architecture (n.d.). Retrieved January 30, 2016, from http://moveit.ros.org/documentation/concepts/

[9] Ashraf Elfasakhany1,2, Eduardo Yanez2, Karen Baylon2, Ricardo Salgado2 , 1 Department of Mechanical Engineering, Faculty of Engineering, Taif University, Al-Haweiah, Saudi Arabia , 2 Tecnológico de Monterrey, Campus Ciudad Juárez, Ciudad Juarez, Mexico "Design and Development of a Competitive Low-Cost Robot Arm with Four Degrees of Freedom ", Modern Mechanical Engineering, 2011, 1, 47-55 , doi:10.4236/mme.2011.12007, Published Online November 2011 (http://www.SciRP.org/journal/mme) from http://docs.ros.org/indigo/api/pr2_moveit_tutorials/html/planning/src/doc/perception_configuration.html

[10] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss. Wolfram Burgard , "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees ", Autonomous Robots (2013) , Preprint, final version available at DOI 10.1007/s10514-012-9321-0