Using the Robot Operating System (ROS) To Operate the Calliope2SP

By Gwendolyn Tennell

1. Introduction

Robot research needs more students interested in robotics to continue revolutionizing the industry. Depending on the environment and the need; creating robot software is not easy. Building robots is all about timing, doing several things at once, and about interfacing with an active and changeable world. Robots do not have the senses of hearing, touch or vision. They do not know how to solve an unexpected problem or feel an object. Neither do they have the ability to adapt to situations on their own. Therefore, robots must be programmed. Programming robots involves the monitoring and manipulation of services that often occur at the same time. The Calliope2SP was designed by the Tekkotsu Lab at Carnegie Mellon University. It is an Asus netbook which is mounted on an iRobot Create (see figure 1). The Calliope2SP is a good robot to use within robotic research, as well as robotic competitions, because of its ability to do multiple maneuvers.

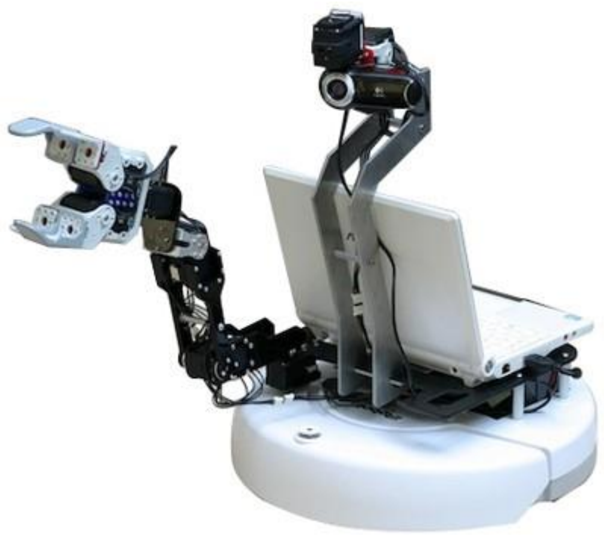


Fig. 1

Robotic competitions are events in which students program a robot to compete against other robots to achieve a goal. Robot competitions provide challenges and encourage research in autonomous technology. To program a robot for competition, a developer considers the number of joints and their geometric arrangement to dictate an intended task. The speed, size and load capacities are also considered (Craig, 11). Robotic competition challenges that look on purpose, rather than competition, and on autonomous, rather than not, are showing promise (Anderson, 2).

2. The Robot Operating System

The open source Robot Operating System was released in 2013. Its framework offers a collection of developer tools, libraries, and conventions used across wide robotic platforms. The aim is to simplify the task of creating complex autonomous robot behavior. Although training in ROS can be difficult, it is a free platform which encourages shared development. It provides tools and libraries for the beginner, as well as the advanced programmer. Programmers may use the Robot Operating System, a Linux-based environment, to work collectively with programmers on other servers, such as Microsoft Windows, and Mac OS X. There is also diversity with Rosjava, the native Java ROS client library, which has empowered ROS-based software to be written for the Android OS; and Roslibjs, a JavaScript client library, which has been developed to integrate software into the Robot Operating System through any standard web browser (Robot, par 3). The Robot Operating System enables the sharing of coded algorithms between researchers by providing a shared architecture on which to run them (see figure 2). ROS also provides beginning and advanced tutorials, and the Turtlesim simulation for training in robotics.

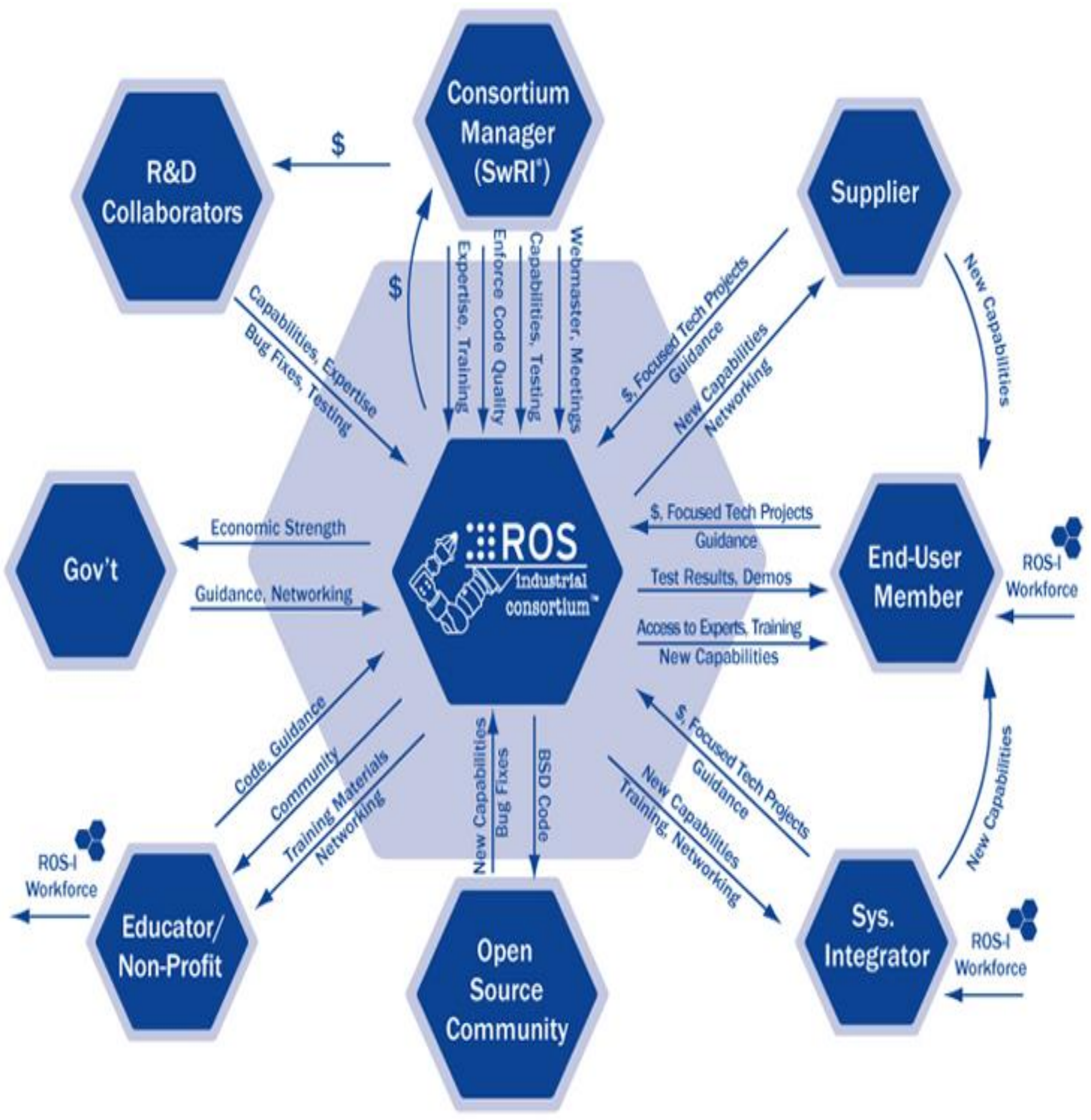


Fig.2

The Robot Operating System makes it easy to incorporate new hardware and external libraries; it uses a shared structure to swiftly tie in new modules. ROS is versatile, and can integrate C++, Python, Java, Lisp, or Octave based codes. Since students have broad varying qualifications, ROS is the industrial strength choice for some researchers. Larger numbers of research groups are creating code targeted at ROS (Thread, 2). The Robot Operating System is complex, but scalable and autonomous programming is accessible. Since coding may sometimes be frustrating, my approach is to offer a starting point with pre-coded C++ ROS packages for the Calliope2SP. The packages may reduce some of the complexities of using ROS, and still allow for growth. My research corrected the errors faced while building the workspace for the Calliope2SP packages in ROS. As they advance in skill, users can leverage the large libraries of existing ROS modules to expand their knowledge of operating the robot.

3. Related Work

An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs. The operating system is an important factor of the system software in a computer system. Application programs typically need an operating system to function (Operating, 1). The diagram in figure 3 displays how the

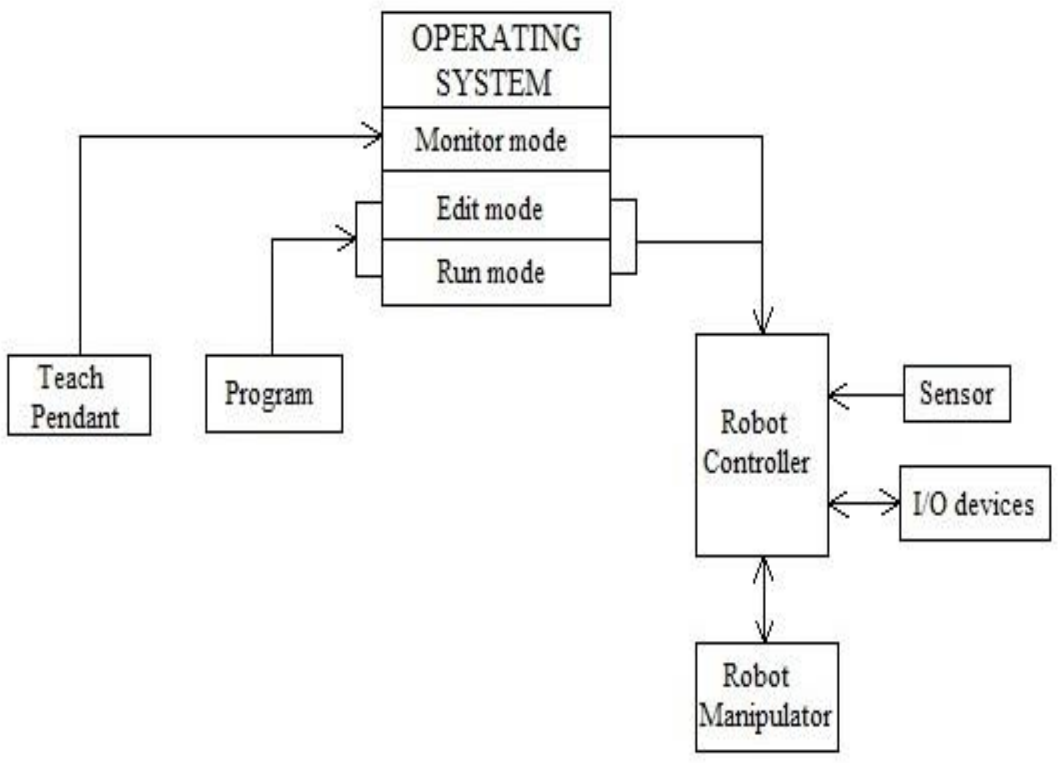
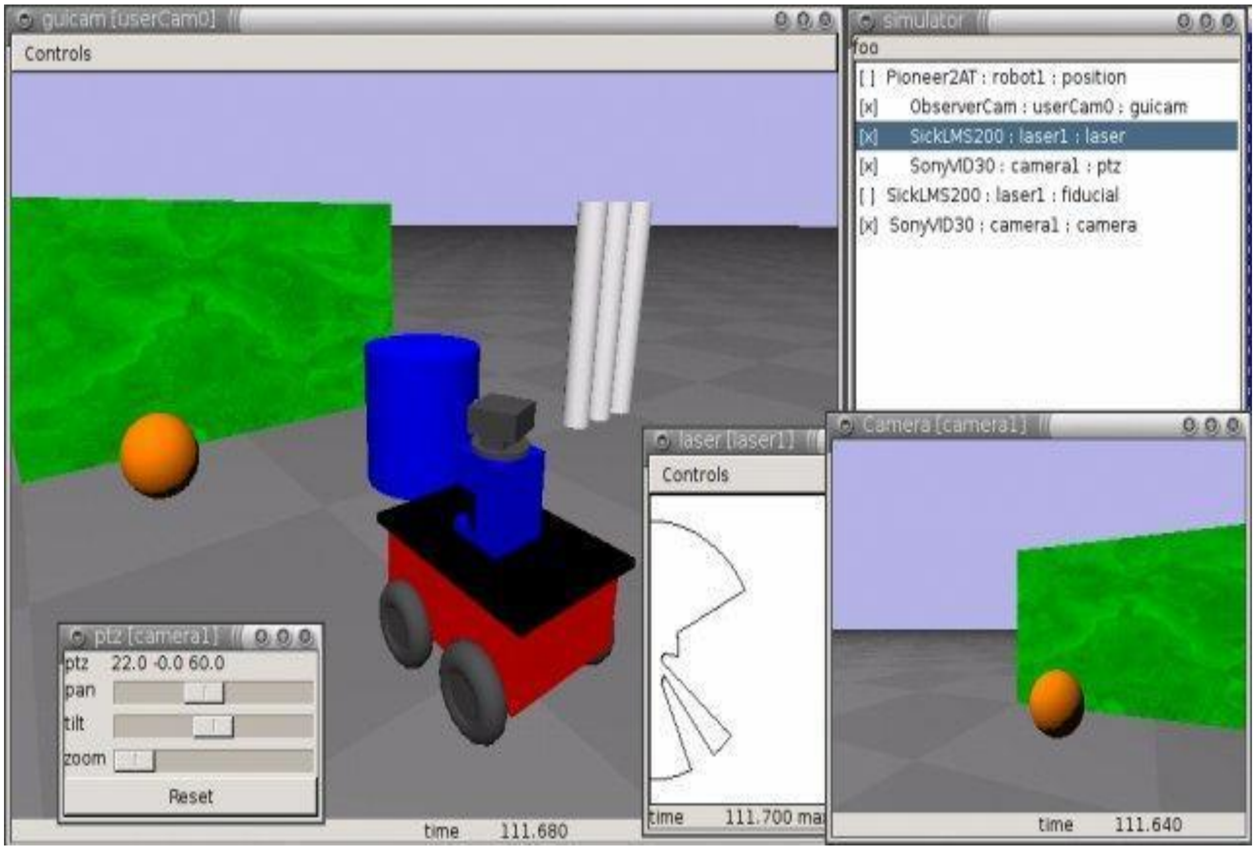


Fig. 3

OS, programs, and the robot connect with each other. I have chosen 3 commonly used platforms, Player Project, Microsoft Robot Developer Studio (RDS), and Tekkotsu to compare to the Robot Operating System. All are open source and support robotic research. They also allow for code to be written in multiple languages, and can operate across different processors. Like ROS, they all have beginning and advanced tutorials for operating robots.

3.1. Player Project



[ ] Pioneer2AT : robot1 : position
[x] ObserverCam : userCam0 : guicam
[x] SickLMS200 : laser1 : laser
[x] SonyVID30 : camera1 : ptz
[ ] SickLMS200 : laser1 : fiducial
[x] SonyVID30 : camera1 : camera

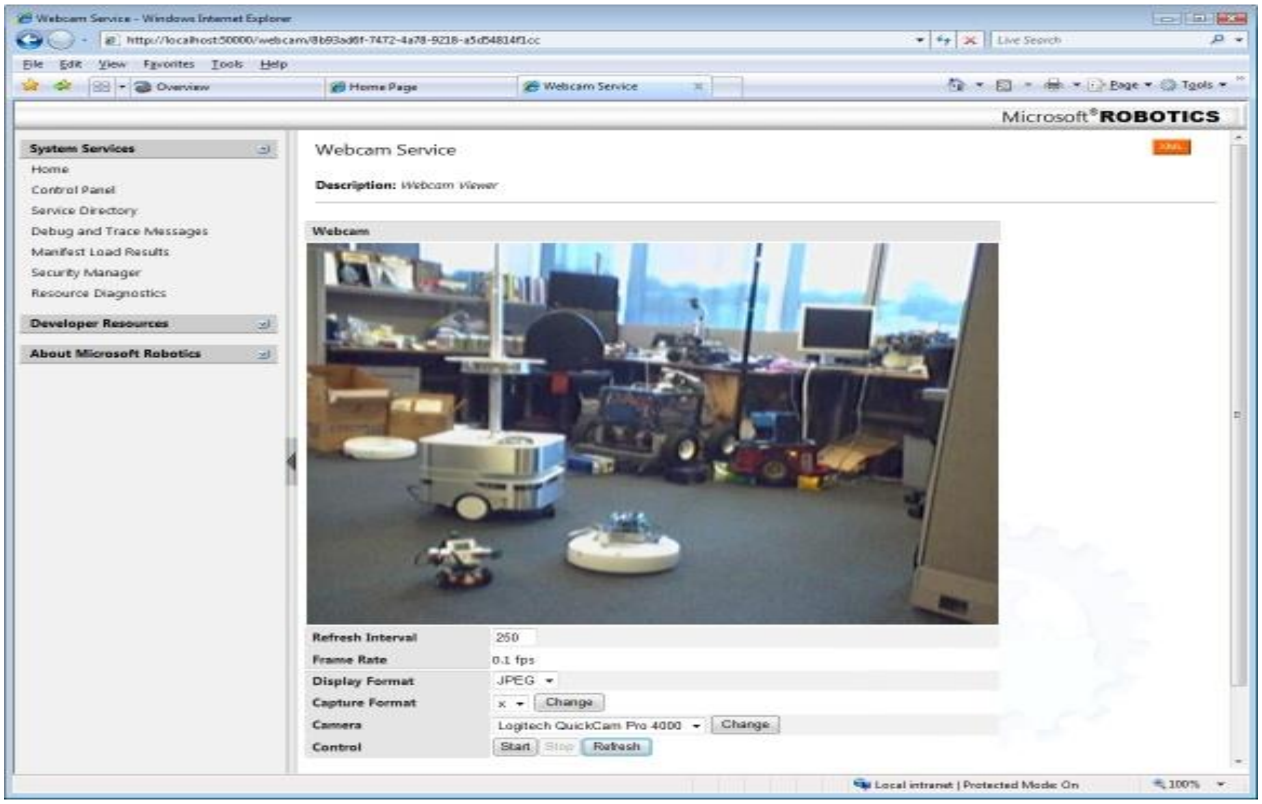
ptz 22.0 -0.0 60.0

Fig. 4

Simulation is a common way to begin programming in robotics, and Player/Stage project is a robot interface that is often used (Player, par. 1) (see figure 4). The Player Stage simulation, which has been around longer than Microsoft Robot Developer Studio, Tekkotsu or ROS, allows focus on algorithms. Therefore students may learn software without the hardware issues. This is a good starting point, and Anderson believes that if a student decides to enter robotic competitions, Player/Stage can also be used with low and high cost platforms (Player Stage, 1).

3.2. Microsoft Robotics Developer Studio

Microsoft Robotics Developer Studio, released in 2006, provides a window-based interface for programming a wide range of robots. Hobbyists controlling toy robots and serious roboticists controlling complicated robots can use the same software development kit (SDK). The central core supports distributed processing and does not require any specific robotic OS. The application resides on a Web server and the robot communicates with the application using a wireless or wired interface (Rea, par. 4). Figure 5 depicts real-time monitoring of robotics sensors. The main programming language of the Robotics Studio is C#, and it also provides a simulation environment.



Refresh Interval	250
Frame Rate	0.1 fps
Display Format	JPEG
Capture Format	x
Camera	Logitech QuickCam Pro 4000

Fig. 5

3.3. Tekkotsu

Tekkotsu was released in 2010. The Tekkotsu framework offers a collection of tools, a GUI interface, and can share C++ objects across processors (Tira-Thompson, 2). Tekkotsu uses Mirage simulation and the Chiara robot, as shown in figure 6, as is an excellent starter platform for robotics programming. Because Tekkotsu runs a massive 200 megabyte library file plus a 30 megabyte executable, it is much slower to develop low-level code than ROS. Because of its speed, it is less desirable as a general research tool than ROS, but it is a good teaching tool.

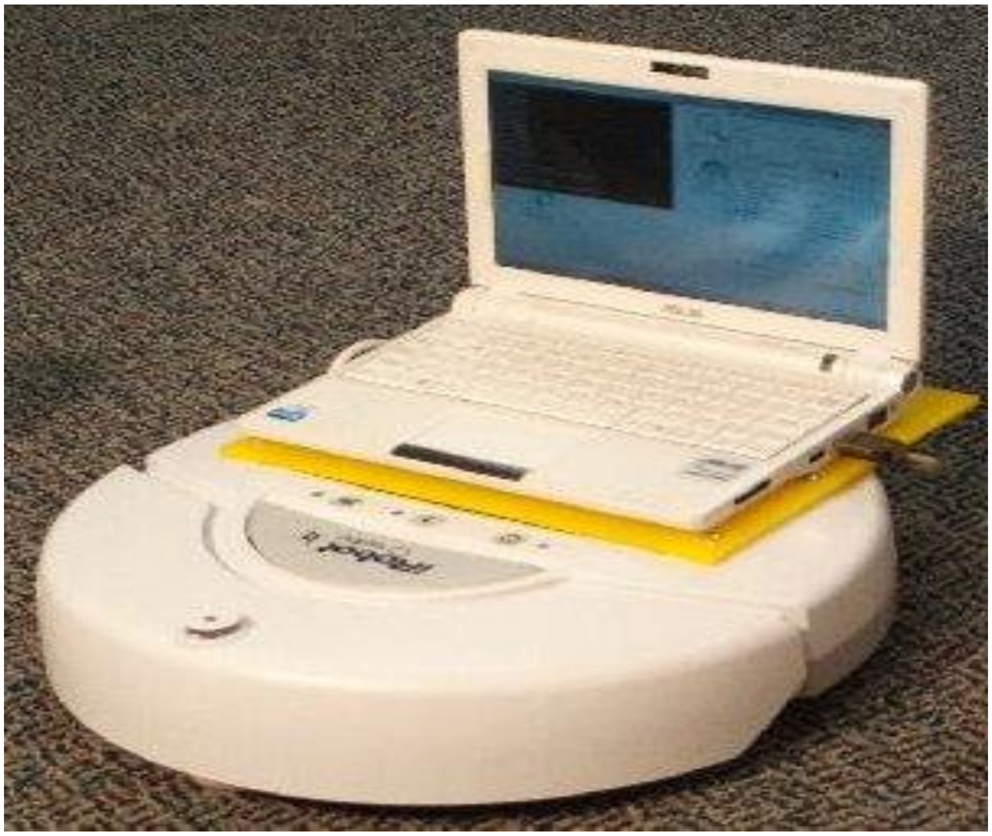


Fig. 6

4. Calliope 2SP Packages

Five packets are needed to maneuver the kinematics controlled robot, Calliope2SP, in the Robot Operating System. Kinematics control is “the coordination of the links of a kinematics chain to produce the desired motion of the robot” (Anh, 1). The Actuator Array Driver performs standard works, such as subscribing and publishing to a command topic, and parsing joint limits. The Calliope package houses the launch, the urdf and the yaml files. The Calliope Driver contains the KinematicSolver header file and the C++ codes for nodes and demos. The Dynamixel Array allows any number of controllers to add new behaviors to the robot. Finally, the Dynamixel Driver package provides low-level IO for the Dynamixel servos. Each packet contains a CMakeLists.txt, where the programmer configures the packages to build on. For example, add or target libraries and executables, find other dependent packages, or create messages. Each packet also contains a package.xml, which list dependencies. The packets are built into a catkin workspace in ROS.

5. Discussion and Conclusion

Through this research I found that all of the robot platforms were able to perform most of the same applications. The Robot Operating System had the edge because of its ability to use a shared structure to quickly tie in new modules. By creating the tutorial www.calliope2sptutorial.weebly.com, I may provide a stress-free foundation to introduce undergraduates to robotics. This tutorial not only produces a vehicle in which to introduce a novice to robotics, but may also prepare an undergraduate for challenging robot competition events. The code in the tutorial can be built upon to move the Caliope2SP autonomously. A beginning Calliope2SP tutorial may encourage and increase an undergraduate’s confidence level towards a fun and interesting journey in programming robots. I recommend that this research is continued, if not by me, by an individual or a group that would finish and build the Calliope2SP tutorial into the Robot Operating System.

6. Acknowledgments

I would like to thank Dr. Monica Anderson for her help as a mentor. I would also like to thank her for the use of the Calliope2SP robot and the robotics lab at the University of Alabama. I would like to thank Connor Lawson for access to his research files. I would like to thank the Distributed Research Experiences for Undergraduates (DREU) and the National Science Foundation (NSF). This work is supported in part by DREU and the NSF grant CNS-1303156

7. References

1. Anderson, M.; Jenkins, O.C.; Osentoski, S., "Recasting Robotics Challenges as Experiments [Competitions]," in Robotics & Automation Magazine, IEEE, Vol 18, No 2, (2011).

2. Anderson, M.; Thaete, L.; Wiegand, N., "Player/Stage: A Unifying Paradigm to Improve." *Department of Computer Science* [The University of Alabama]: 1-5.

3. Anh, Ho Pham Huy, and Nguyen Thanh Nam. "Novel Adaptive Forward Neural MIMO NARX Model for the Identification of Industrial 3-DOF Robot Arm Kinematics." *Intech*. N.p. Web. 11 Nov. 2014

4. Craig, John J., *Introduction to robotics: mechanics and control*. 2nd ed. Reading, Mass.: Addison-Wesley, 1989. Print.

5. "Operating System." *Wikipedia*. Wikimedia Foundation, 11 Feb. 2014. Web. 8 Nov. 2014.

6. "The Player Project." *Player Project*. Web. 8 Nov. 2014.

7. Rea, Sara. "An Introduction to Programming Robots with Microsoft Robotics Studio." *An*

Introduction to Programming Robots with Microsoft Robotics Studio. N.p., n.d. Web. 10 Sept. 2014. http://www.devx.com/dotnet/Article/32729.

8. "Robot Operating System." *Wikipedia*. Wikimedia Foundation, 11 Jan. 2014. Web. 8 Nov. 2014.

9. "Thread: Tekkotsu vs. ROS." *Trossen Robotics Community RSS*. N.p., n.d. Web. 18 July 2014.

10. Tira-Thompson, E.; Touretzky, D.S., "The Tekkotsu robotics development environment," *Robotics and Automation (ICRA), 2011 IEEE International Conference on* , vol., no., pp.6084,6089, 9-13 May 2011.