ROS as an Undergraduate Project-based Learning Enabler

Dr. Stephen Andrew Wilkerson P.E., York College of Pennsylvania

Stephen Wilkerson (swilkerson@ycp.edu) received his PhD from Johns Hopkins University in 1990 in Mechanical Engineering. His Thesis and initial work was on underwater explosion bubble dynamics and ship and submarine whipping. After graduation he took a position with the US Army where he has been ever since. For the first decade with the Army he worked on notable programs to include the M829A1 and A2 that were first of a kind composite saboted munition. His travels have taken him to Los Alamos where he worked on modeling the transient dynamic attributes of Kinetic Energy munitions during initial launch. Afterwards he was selected for the exchange scientist program and spent a summer working for DASA Aerospace in Wedel, Germany 1993. His initial research also made a major contribution to the M1A1 barrel reshape initiative that began in 1995. Shortly afterwards he was selected for a 1 year appointment to the United States Military Academy West Point where he taught Mathematics. Following these accomplishments he worked on the SADARM fire and forget projectile that was finally used in the second gulf war. Since that time, circa 2002, his studies have focused on unmanned systems both air and ground. His team deployed a bomb finding robot named the LynchBot to Iraq late in 2004 and then again in 2006 deployed about a dozen more improved LynchBots to Iraq. His team also assisted in the deployment of 84 TACMAV systems in 2005. Around that time he volunteered as a science advisor and worked at the Rapid Equipping Force during the summer of 2005 where he was exposed to a number of unmanned systems technologies. His initial group composed of about 6 S&T grew to nearly 30 between 2003 and 2010 as he transitioned from a Branch head to an acting Division Chief. In 2010-2012 he again was selected to teach Mathematics at the United States Military Academy West Point. Upon returning to ARL's Vehicle Technology Directorate from West Point he has continued his research on unmanned systems under ARL's Campaign for Maneuver as the Associate Director of Special Programs. Throughout his career he has continued to teach at a variety of colleges and universities. For the last 4 years he has been a part time instructor and collaborator with researchers at the University of Maryland Baltimore County (http://me.umbc.edu/directory/). He is currently an Assistant Professor at York College PA.

Dr. Stephen Andrew Gadsden, University of Guelph

Andrew completed his Bachelors in Mechanical Engineering and Management (Business) at McMaster University in 2006. In 2011, he completed his Ph.D. in Mechanical Engineering at McMaster in the area of estimation theory. Andrew worked as a postdoctoral researcher at the Centre for Mechatronics and Hybrid Technology (Hamilton, Ontario, Canada). He also worked as a Project Manager in the pharmaceutical industry (Apotex Inc.) for three years. Before joining the University of Guelph in 2016, he was an Assistant Professor in the Department of Mechanical Engineering at the University of Maryland, Baltimore County. Andrew worked with a number of colleagues in NASA, the US Army Research Laboratory (ARL), USDA, NIST, and the Maryland Department of the Environment (MDE). He is an ASME and IEEE member, and a Professional Engineer. Andrew was an Associate Editor for the Transactions of the Canadian Society for Mechanical Engineers and is a reviewer for a number of ASME and IEEE journals and international conferences. Andrew is a 2018 Ontario Early Researcher (ERA) award winner (on intelligent condition monitoring strategies), and has been nominated for the 2018 University of Guelph Faculty Association (UGFA) Teaching Award.

Mr. Andrew Lee, University of Guelph

Lee has applied his mechanical engineering knowledge and STEM teaching skills as a camp assistant in the Take Flight Robotics program, a summer experience designed to engage and inspire high school interested in STEM fields. In 2015, Lee helped participants build and program their own small drones, and in 2016 he developed a new program to help them modify remote controlled cars using coding skills and microcontrollers.

Mr. Robert Nicholas Vandemark
Miss Elyse Hill,

I am a first year PhD student with current interests in control and estimation theory and pedagogy research. I hope to obtain a faculty position in mechanical engineering post-PhD and combine my interests into a new field of research.

Ms. Amy Domenique Gadsden, University of Alberta

Amy Domenique completed her Bachelor of Fine Arts at Mount Allison University (New Brunswick) and a Bachelor of Education at Nipissing University (Ontario) in 2012. Following this, she completed a Master of Education at Nipissing University in Special and Inclusive educational praxis in 2014. Amy Domenique is a certified teacher and has worked as an elementary and secondary classroom teacher in both the province of Ontario and Alberta. She is a member of the Ontario College of Teachers and the Alberta Teachers Association. She is an experienced university instructor, guest speaker, and graduate teaching assistant. Amy Domenique has also worked as a research assistant in the JP Das Centre for Developmental and Learning Disabilities (Alberta) as well as the Western Canada Centre for Deaf Studies (Alberta). Concomitant with her relevant professional working experience, she is a member of a variety of organizations that promote and advocate for the inclusion of students with special needs. Amy Domenique is also a Director of the Learning Disabilities Association of Alberta, and a member of its Ontario organization. Currently, Amy Domenique is a doctoral student in the department of Educational Psychology at the University of Alberta, conducting research in Special Education.

ROS as an Undergraduate Project Based Learning Enabler

Future engineering and science jobs will require a greater degree of specialty and diversity at the same time. In manufacturing and service industries robots will likely play a huge job generator. Self driving cars, trucks, and humanoids will only be the start. Advanced robots have traditionally been taught heavily at the graduate level, but not until recently at the undergraduate level. However, the Robotic Operating System (ROS) is a game changer in this regard. ROS allows programmers and engineers to tackle extremely difficult problems without specific knowledge of some of the components. In this paper we look at a year long study of robotic arm mechanisms using a PBL technique. We detail the learning difficulties encountered when developing a program from scratch as well as some of the successes. As part of our measurement of merit, we provide our materials on the internet and track their usage by others. Details of where and how we obtained our data are also provided. The current project is based on the Kobuki Turtlebot and the Trossen Robotics Arm Pincher. In this PBL we attempt to mount a robotic arm on the Turtlebot to retrieve objects located in remote locations using a previously built map. Then building off other student projects we attempt to extend our Kobuki's capabilities from basic navigation to navigation with a mission and purpose.

Introduction

Robotics for the current generation has been a motivator for Project Based Learning (PBL). Since our last report we have streamlined the process rather than expanding and started collaborations with other engineering programs[1,2]. It is no surprise that PBL and Robotics have been used at every level of education to induce student ingenuity and scholarship activities. First Robotics[3,4] and Vex[5,6] leads the way with K-12[7] to undergraduate programs[8]. One of the questions we are trying to answer is at what level of difficulty can we introduce technologies to undergraduate students and have them respond. There are several pit falls in such a program. For example, when the professors do too much of the work on the project, with the students learning, but not doing. Another is where the professors leave the students with little guidance and the tasks required overwhelm and frustrate them. In many cases with ROS and PBL we are exposing students to techniques and methods that are by and large exclusively graduate level materials. Additionally, many of the prior PBL project study exercises are centered on competitive goals. Just the same, this is not a requirement of PBL and the same learning objectives can also be achieved by incorporating them into the design and project requirements. Students like having goals and structure, but can also be motivated by doing something unique and innovative. Ramos and Espinosa[9] used PBL with robotics and showed that the same learning objectives of a traditional classroom could be achieved via PBL alone. The essence of their approach was in building a pupil based learning model using PBL that served as a platform from an instructor based setting to one of self exploration. Self discovery and learning with minimal input has been shown possible by Hees et al[10]. Notwithstanding, students will devote large amounts of time and energy learning new materials to accomplish specific tasks or goals if they feel it is in their best interest.

In this ongoing study we used the above basic principles to examine if; undergraduate students might use ROS nomenclature to master advanced concepts. As stated before our initial fear was in overreaching. In our first semester we managed a balanced approach, while in our second semester we overreached and ended up with unfinished projects and frustrated students. We believe this was due to the professors allowing the students to undertake tasks that were far too ambitious without the proper support. In order to do really difficult projects like the ones imagined by our second semester students, the professors would have needed to intercede and help with the project with more than just advice. While student professor interactions like this are not counter productive, they are largely difficult with more than one or two students especially with faculty who are fully loaded with course work. Graduate studies are the exceptions. Graduate students are more apt to research and formulate solutions to complex problems with lower levels of instruction or outside help. For these projects a balance in the interactions with professor involvement need to be choreographed correctly for success. In our first semester we did this far better than in the second. Nonetheless, other students were still eager to pursue these studies on their own time the following summer and then again in the fall. Being cautious, we were restrictive in our student selection on the ongoing effort. Therefore, this study involves only two students. One in the fall who is now doing a semester long cooperative program for a startup company using robotics for manufacturing and the other in the spring who is picking up where the other left off from the fall of 2016 semester.

It is also important to note that we are operating at a small liberal arts college with an engineering and computer science program that has limited resources. Many programs like the one at the United States Military Academy have excellent resources and in many cases can purchase the solution to portions of their projects making success readily obtainable. When these resources are not available students and professors need to be creative and imaginative. The PBL objectives also need to be realistic and achievable.

ROS has been around for a number of years and is gaining popularity in academics and hobbyist alike. ROS is also used by the mainstream military research groups and is a staple of many graduate programs at larger schools. In 2010 Reid et.al.[11] used the WAMbot to navigate and explore urban environments while performing visual object recognition. While our current project is far less ambitious, the ROS architecture gives us the ability to think big on a relatively small scale. ROS has become a great open source project in the past 5-7 years and it has many examples and tutorials[1]. The original effort involved Stanford University efforts to include artificial intelligence into an in-house software systems that could be used for robotics. After that a small start up company called Willow Garage provided essential resources extending the initial idea advanced at Stanford. The project has continued to grow and now has contributions from numerous researchers from around the world. The basic functionality of ROS facilitates difficult projects by building off of what others have done. The system level problems that might be encountered with new hardware, interrupts and other software hardware issues are handled by ROS enabling students to focus on capabilities rather than the details of hardware integration. Additionally there are numerous books available that explain and teach ROS. We initially employed Kane's "A gentle introduction to ROS, independently published (2013).[12]" Subsequently, we've started using Goebel's "ROS Robotics by Example[13]" and "ROS by Example A Do-It-Yourself Guide to the Robotic Operating System[14]." To help with some of the vision work we also employed Rosebrock's "Practical Python and OpenCV.[15]"

One of our goals of this effort was to motivate students to do the extra work required to use these advanced topics in robotics using PBL. Our assessment criteria is not meant to be absolute. As pointed out by Prince[16] PBL assessments are subjective at best. Therefore, we dispensed with the typical student surveys and other data collection techniques and focused on the project outcome and completion as a measure of merit. Since our sample size is so small it would be statistically insignificant. Our program included both male and female students with no real observable differences in motivation. Since the current participants are selected from a small cadre of undergraduate students, most of which are likely to go on to graduate studies, our sampling is admittedly skewed. A larger question that might be asked in future studies is would this work with relatively unmotivated students and act as a catalyst for improved studies and course work in other areas. For now, we hold our efforts to a small motivated group of students.

Technologies

The technologies hardware and software used in this program are briefly reviewed. In most cases, these required some initial work for the students as they were unfamiliar with the common uses and operations of these items. By and large most of the students use a Microsoft

1 ROR Tutorials can be found at: http://wiki.ros.org/ROS/Tutorials and include everything from visualization to interfacing with Arduinos and other electronic packages software and hardware.

(MS) based Personal Computer (PC) system with a few exceptions where students use Apple based systems. While the Apple systems are Unix-based and that makes this easier, using these systems created a whole new series of mistakes, problems and other difficulties. To avoid all of the issues faced using varying systems we went to a single platform for our base station, namely the Intel Nuc[2] with Ubuntu 14.04[3] installed on it. Fortunately, Ubuntu is the same OS and version we are using with the Kobuki Turtlebots. This eliminates the endless issues that occur with virtual systems loaded on MS PC systems as well as those on the MAC system. Furthermore, we use the ROS version indigo on both systems thereby eliminating other issues that might arise using various versions of ROS on different platforms. This forces students to learn Ubuntu Linux, but results in less problems with communications and switching between different systems. In other words, it eliminates one of the complexities in doing this type of work and reduces student *"I'm Stuck"* time. Figure 1 Shows the Nuc-I7 system used for much of this study.

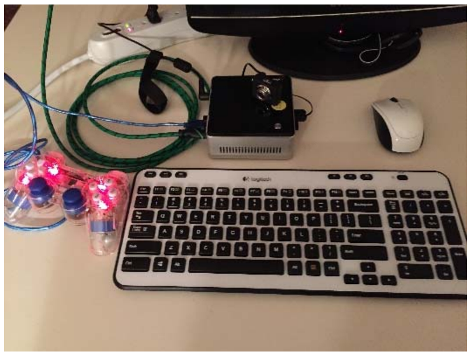


Figure 1. Nuc-I7 system with Ubuntu 14.04 OS.

For our mobile platform we chose the Kobuki Turtlebot. The Kobuki is a relatively low-cost mobile research base. It was designed for education and research using ROS. It has a remarkably accurate odometry system along with a calibrated gyroscope that enables reasonably good navigation. The Kobuki base includes a 2200mAh battery pack, a Kinect sensor, an Asus 1215N laptop that has a dual core processor4. Our project employed Ubuntu 14.04 on this system and students were forced to learn how to build an Ubuntu system for use with the Kobuki systems and were therefore able to work on both systems with ease. One of our Kobukis is shown below in Figure 2.

2 The Intel® NUC is a powerful 4x4-inch mini PC productivity features, including a customizable board that is ready to accept, memory, storage, and various operating systems: https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html

3 Ubuntu is a computer operating system based on the Debian Linux distribution. The system is open source and distributed as free for use on desktop devices. It is named after the Southern African philosophy of Ubuntu meaning "Humanity Towards Others"

4 The Kinect is based on the PrimeSense's technology. Prime sense was an Israeli 3D sensing company based in Tel-Aviv and was bought by Apple inc in 2013. PrimeSense is known due to its licensing of the hardware and chips used in Microsoft's Kinect motion-sensing system: https://en.wikipedia.org/wiki/PrimeSense

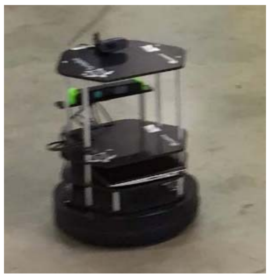
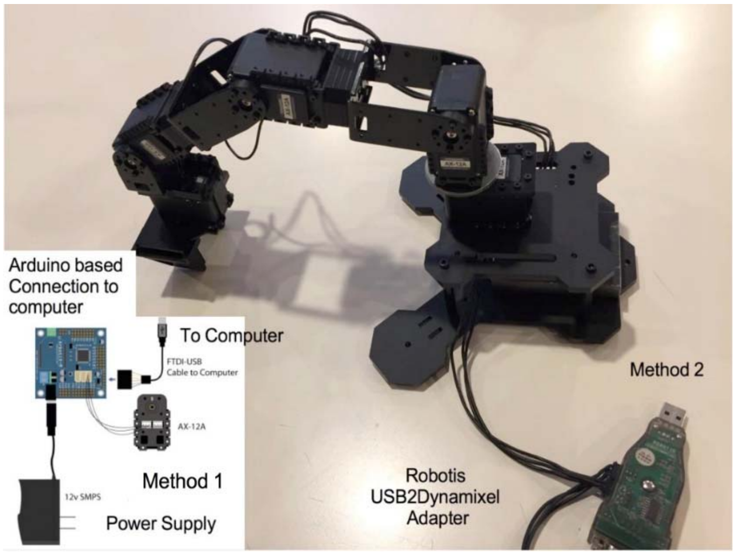


Figure 2. Kobuki Turtlebot 2 with Asus laptop and Kinect sensor.

The Turtlebot has a suite of "Getting Started" tutorials for learning ROS and the Turtlebot system[5]. In fact the tutorials alone could keep students busy learning the Turtlebot system for a semester or more. Included are some basic learning lessons with the Tutlebot along with challenges and sketches to teach how to do the work in simulation. This allows small schools with limited resources to tackle difficult problems without a large grant or a lot of hardware. Our program benefited from the fact that we already had 3 Turtlebots and some of the associated hardware to start. However, this would not have precluded us from doing many of our experiments in simulation mode with the same challenges and outcomes.


Arduino based Connection to computer
To Computer
FTDI-USB Cable to Computer
AX-12A
Method 1
12v SMPS
Power Supply
Robotis USB2Dynamixel Adapter
Method 2
Figure 3. Trossen Robotics Arm "PhantomX Pincher".

We chose the Trossen Robotics arm "PhantomX Pincher" to start. The PhantomX Pincher uses the AX-12 servo motors. The motors allow each successive motor to be daisy chained together with a power rail and one single signal wire. This greatly simplifies the wiring. With the use of a series of hexadecimal addresses individual servos functions can be actuated or used simultaneously in combination with other servos[6].

5 Tutorials for the Turtlebot can be found at: http://learn.turtlebot.com/
6 Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel: http://support.robotis.com/en/product/actuator/dynamixel/ax_series/dxl_ax_actuator.htm

The Pincher robot arm is a 5 degree-of-freedom arm and an easy addition to the Turtlebot ROS robot platform. The arm kit comes with everything needed to physically assemble and mount the arm as a stand-alone unit or as an addition to your Turtlebot platform. Shown in Figure 3 is one of our two robotic arms set up without the Turtlebot platform.

Each of the servos in the chain needs to have an assigned unique id and this can be accomplished using the Dyna-Manager software provided by Trossen robotics[7]. Due to the many power software connection permutations available for the arm, this proved to be quite frustrating for both the students and faculty. In the end we used method B shown above in Figure 3. It is also possible to use method A and that connection is shown for completeness in Figure 3 on the left. While the arm comes with some software that allows it to be manipulated using a GUI interface, it was our desire to write our own software and to control it within the ROS framework. To that end we used Python and the *PyPot* libraries. The PyPot libraries handles the communication with dynamixel motors from Robotis. The connection uses a USB USB2DYNAMIXEL communication device (See figure above). Specifically *PyPot* libraries allowed easy access (both reading and writing) to the different registers of any dynamixel motors. The registers include data values giving position, speed, and torque. There are a whole list of registers that can be directly accessed and a manual can be found on the Robotis website[8]. Doing this at the lowest level would have been a project within itself and we therefore relied on the *PyPot* libraries.

However, what proved totally invaluable was the *PyPot* tutorials[9]. These allowed the opening and closing of connections to the dynamixels as well as the movement and sensing of the corresponding dynamixel's position and condition using a user written python program. This is done using low-level API commands. Using these *PyPot* libraries one can get or set a value to a motor by directly sending a request. The initial setup took sometime to initiate, ~10 seconds, but once done the arm's servos could be accessed in real time from that point on. However, the best way to pass a request to the arm is using a list rather then individual servo requests. A list of servo commands will be executed simultaneously and this takes the same amount of time as getting or sending information to a single servo. Nonetheless, it is necessary to experiment with these libraries prior to using them as there are many nuances in the use of these arms.

Preliminary Results

Using these methods we were able to get the arm to pick up small objects that were placed in front of the arm at known distances. The long range goal would be to get the arm to do this using a vision based system, but for now we restricted ourselves to using geometry based movements and inverse kinematics. To simplify the process we initially considered the Pincher arm as a 2 Degree of Freedom (DOF) system. Figure 4 shows the basics of this operation. Once the object location is known the lengths of segment 3 and y1 can be found. Using the law of Cosines angles α, β, and γ can be found and the hand at the end of the arm can be located. Initially the arm is in the vertical direction where λ and ζ are zero. Now we add one more DOF into the equation in Figure (b) by adding the pincher pivot. We could have just allowed the 4th

7 Trossen Robotics: http://www.trossenrobotics.com/
8 Robotis Website: http://support.robotis.com/en/product/dxl_main.htm
9 *PyPot* Tutorials can be found at: http://poppy-project.github.io/pypot/tutorial.html

motor to remain in the zero position thereby extending the length of segment 2. However, we turned the 4th servo down in the vertical direction requiring some more calculations as shown in Figure (c) thus requiring the recalculation of all of the angles and lengths. Finally, if we form a 180 degree semicircle in front of the arm we can add our 4th DOF to the equations on the first servo in the arm. By knowing the distance from the center of rotation of the first servo to the object, the arm's movements can be choreographed to reach down and pick up the object by specifying the absolute angles of each servo motor from the distance and angular location of the object. The 5th degree of freedom is the pincher at the end of the arm that picks up the object. This initial effort will prove beneficial when we add a vision based system in later student based PBL scenarios. A video of the arm in operation can be found on you tube at: goo.gl/od6KzB

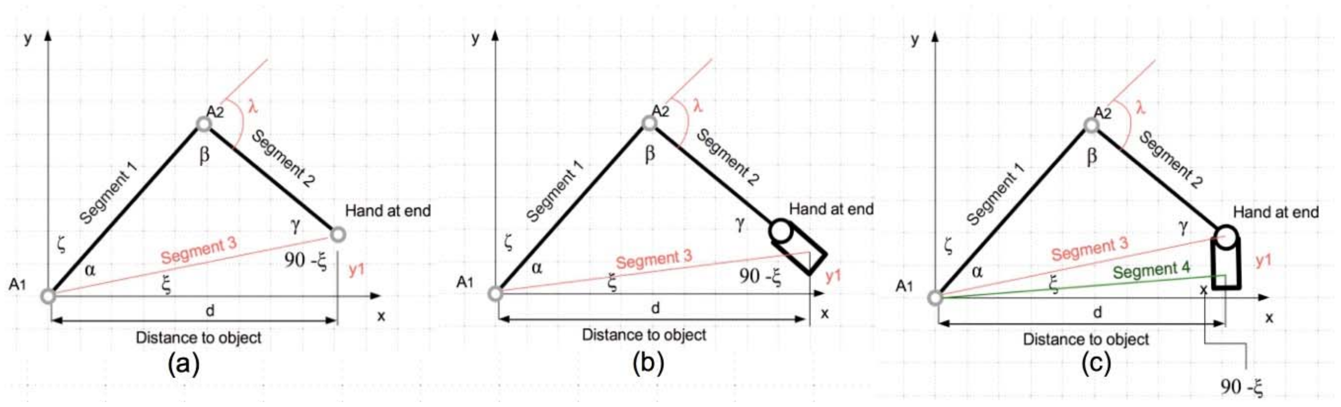


Figure 4. (a), (b), and (c) Robot Arm Kinematics.

In order to facilitate efforts elsewhere a snippet of python code that moves all of the servos in the arm into the zero position is provided in Appendix A. Using this snippet of code or code provided in the PyPot tutorials it's possible to examine all of the formats thereby reducing errors in passing information to the PyPot libraries.

Now we need to ROSify the python code and then test that in the ROS environment. We performed that and then tested the code by adding a joystick to move the arm. The objective is to hold different buttons on the joystick while moving the sticks to reposition different servos on the arm. We used a standard PS2 joystick[10] and the buttons 1 through 4 on the top and the right joystick and then one of the back buttons and the left joystick for the pincher. The code has been posted on GitHub at: goo.gl/ZJJXaj. This will allow other efforts to use what we have created and further improve upon the methods and techniques. It also offers a good starting point for a new PBL student project.

10 Playstation 2 Accessories: https://en.wikipedia.org/wiki/PlayStation_2_accessories

Future Work and Activities

Following this initial effort we will incorporate the arm onto the Kobuki Turtlebot. Then combining several sub projects already completed the robot will navigate to a specific room in the Engineering building and then using ArUco tags locate an object and attempt to retrieve it. The ArUco tag library has been developed by the Ava group of the University of Cordoba in Spain. The library provides real time marker based 3D pose estimation. This allows the robot to obtain additional information about its location relative to the tag. Figure 5 shows the tags and the resulting vector information that can be obtained using these modules. In previous efforts we used ArUco tags on a moving robot and then used that information to create a leader follower algorithm. In Figure 5, the Aruco tag ID is identified and the Cartesian coordinates (x, y, and z) are identified thereby allowing the robot to reposition itself based on this information.

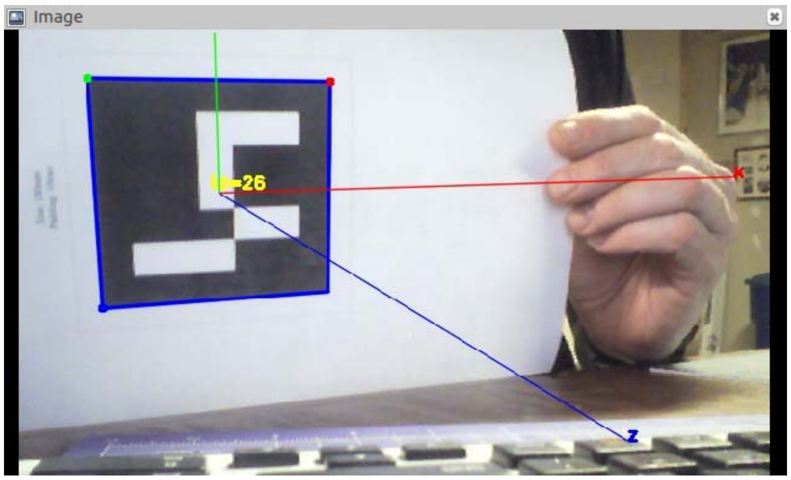


Figure 5. ArUco tag detection in ROS.

There are already several examples of ArUco tag detection on the web and the student was able to recreate these within the ROS environment[11]. The process uses OpenCV libraries and preexisting routines to find the tag and approximate its orientation. Additionally since the student had prior knowledge of using known maps to navigate the combination of the existing programs make it possible to develop unique capabilities. Figure 6 shows an example of using a Kobuki generated map to navigate to a known location. The current arm now works with a joystick and a user can pick and place objects. The Github repository can be found at: https://github.com/csperbeck6/dyna_arm.

Currently, we are using the OpenCV[12] libraries to locate colors using a small camera. To extend the arms capabilities we will have the arm pick and place objects in accordance with colors autonomously. This can be extended to include RF tags or other object indicators.

11 ArUco tag detection: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html Ros and ArUco tags: http://wiki.ros.org/aruco
12 OpenCV is an invaluable source of visual tools: https://www.learnopencv.com/install-opencv3-on-ubuntu/

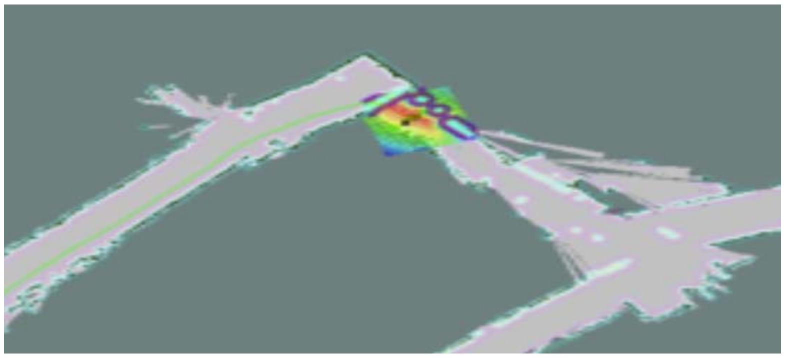


Figure 6. Kobuki generated map for autonomous navigation.

Conclusions

The most difficult portion of these complex PBL experiments is getting the students up to speed in specific areas in order to accomplish tasks. Students are typically required to learn Ubuntu, Linux, ROS, Turtlebot operations, OpenCV, programming languages like Python (they usually know basic C++), and in some cases, all of the above in only two semesters. The simple fact of the matter is that the students will learn a lot, but will also require more assistance than what might be expected in a capstone design. What we found particularly rewarding is that these same students use this to springboard their careers usually into graduate studies or high paying jobs. Nonetheless, the balance between doing too much and not enough remains somewhat elusive. During the course of the summer and fall study there were times when we clearly waited too long before getting involved rather than doing to much. In the ongoing program we intend to be more available and involved in the day to day activities of the students. We also intend to provide more structure in the students studies through the use of specific text books and assignments.

Notwithstanding, these type of PBL assignments, while rewarding, are time-sinks for the student and faculty member alike. The students can not be turned loose with minimal guidance as they will become stuck and frustrated. This has happened on more than one occasion during the past and current semester. What we believe is need based on my personal observations is a more structured approach with supporting materials that will help the student avoid the pitfalls of overreaching. As we continue to develop this approach the need for written materials like the books we have referenced and videos on YouTube are becoming more evident. The benefits of working with these technologies to imbue students with better problem solving skills should be evident. Future engineering students will undoubtedly be task to solve problems where they have little if any formal training. The manufacturing sector, not to mention the service based sector is likely to be filled with robotic system employing automation routines, vision systems, and adapted learning techniques. Engineers will be required to program and fix these systems. Humanoid systems and home robots are right around the corner. Amazon Alexa, and Google home will give way to more sophisticated systems like Jibo and not too far off in the future humanoids like Sophia[13]. Keeping pace with these developments is a daunting task for any modern faculty member. We believe that ROS and technologies that can be comprehended and used by students now will serve them well in the coming years. PBL and student self paced

13 For more information on Sophia see: https://en.wikipedia.org/wiki/Sophia_(robot)

projects may replace the traditional classroom approach as smart systems and robotics become our future teachers. However, for the time being ROS and PBL can be used to motivate, inspire and provide students with problem solving skills that they can use and master. To improve learning we've added 2 additional students for the coming semester. Additionally, we're including a humanoid robot as part of the learning process. Pre and post surveys will be used to help us evaluate student learning and comprehension during the summer semester.

References

1. Wilkerson, S. A., & Forsyth, J., & Korpela, C. M. (2019, June), *Project Based Learning Using the Robotic Operating System (ROS) for Undergraduate Research Applications* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/28768

2. Wilkerson, S. A., & Forsyth, J., & Sperbeck, C., & Jones, M., & Lynn, P. D. (2019, June), *A Student Project using Robotic Operating System (ROS) for Undergraduate Research* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/27515

3. Moylan, William Alexander. "Learning by Project: Developing Essential 21st Century Skills Using Student Team Projects." *International Journal of Learning*15.9 (2008).

4. Barron, Brigid, and Linda Darling-Hammond. "Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning. Book Excerpt." *George Lucas Educational Foundation* (2008).

5. Das, Shuvra, Sandra A. Yost, and Mohan Krishnan. "A 10-year mechatronics curriculum development initiative: Relevance, content, and results—Part I."*IEEE Transactions on Education*53.2 (2010): 194-201.

6. Ruzzenente, Marco, et al. "A review of robotics kits for tertiary education." *Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*. 2012.

7. Grandgenett, Neal, et al. "Robotics and Problem-Based Learning in STEM Formal Educational Environments." *Robots in K-12 Education: A New Technology for Learning: A New Technology for Learning* 94 (2012).

8. George, Sébastien, and Pascal Leroux. "Project-based learning as a basis for a CSCL environment: An example in educational robotics." *First European Conference on Computer-Supported Collaborative Learning (Euro-CSCL 2001)*. 2001.

9. Ramos, Fernando, and Enrique Espinosa. "A self-learning environment based on the PBL approach: an application to the learning process in the field of robotics and manufacturing systems." *International Journal of Engineering Education* 19.5 (2003): 754-758.

10. Hees, Frank, et al. "Developing a PBL-based rescue robotics course." *Proceedings of the First Kuwait Conference on e-Services and e-Systems*. ACM, 2009.

11. Reid, Robert, et al. "Cooperative multi-robot navigation, exploration, mapping and object detection with ROS." *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013.

12. O'Kane, J. M. "A gentle introduction to ROS." Independently published (2013).

13. Goebel, R. Patrick. *ROS by example*. Lulu.com, 2015.

14. Goebel, R. Patrick. *ROS by Example: A Do-it-yourself Guide to the Robot Operating System: a Pi Robot Production*. Lulu. Com, 2012.

15. Rosebrock, A. "Practical Python and OpenCV." *Miami: pyimageseach* (2016).

16. Prince, Michael. "Does active learning work? A review of the research." *Journal of engineering education* 93.3 (2004): 223-231.

17. J. Kim, K. Chang, B. Schwarz, A. S. Lee, M. Al-Shabi, and S. A. Gadsden, *Dynamic Model and Motion Control of a Robotic Manipulator*, Journal of Robotics, Networking and Artificial Life, Vol. 4, No. 2, pages 138-141, 2017.

18. J. Goodman, J. Kim, A. S. Lee, S. A. Gadsden, and M. Al-Shabi, *A Variable Structure-Based Estimation Strategy Applied to an RRR Robot System*, Journal of Robotics, Networking and Artificial Life, Vol. 4, No. 2, pages 142-145, 2017.

Appendix A

```
#  This code snippet is a modification of the code foundation in the PyPot Tutorial:
#  Import Required Libraries
import itertools
import pypot
import numpy
import time
import pypot.dynamixel
# Get the Port with the Dynamixel, not required, but helps simplify things by getting the port **
if __name__ == '__main__':
    ports = pypot.dynamixel.get_available_ports()
    print 'available ports:', ports
    if not ports:
        raise IOError('No port available.')
    port = ports[0]
    print 'Using the first on the list', port
#
# Declare dxl_io as type pypot.dynamixel This usually takes the most time. **
#
    dxl_io = pypot.dynamixel.DxlIO(port)
    print 'Connected!'
#
# Check the Servo ids Also not required
#
    found_ids = dxl_io.scan()
    print 'Found ids:', found_ids
#
# Check to see if we have 5 servos not required...
#
    if len(found_ids) < 5:
        raise IOError('You should connect at least two motors on the bus for this test.')
    ids = found_ids[:5]
    print ids
    #
    #   Enable servos and set the speed
    #
    dxl_io.enable_torque(ids)
    speed = dict(zip(ids, itertools.repeat(100)))
    dxl_io.set_moving_speed(speed)
    #
    # Move all of the Servo to the zero position
    #
    pos = dict(zip(ids, itertools.repeat(0)))
    dxl_io.set_goal_position(pos)
    print 'Done'
```