A ROBOTICS FRAMEWORK FOR SIMULATION AND CONTROL OF A 3D PRINTABLE ROBOTIC ARM FOR USE IN HIGHER EDUCATION

by

Craig Christensen

A THESIS PROPOSAL

Presented to the Faculty of

The School of Computing at the Southern Adventist University

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science Embedded Systems

Under the Supervision of Tyson Hall

Collegedale, Tennessee

May, 2016

A ROBOTICS FRAMEWORK FOR SIMULATION AND CONTROL OF A 3D PRINTABLE ROBOTIC ARM FOR USE IN HIGHER EDUCATION

Craig Christensen, M.S.

Southern Adventist University, 2016

Adviser: Tyson Hall, Ph.D.

Robotic arms have been in common use for a several decades now in many areas from manufacturing and industrial uses to hobby projects and amusement park rides. However, there have been very few attempts to make an inexpensive robot arm with a software stack for use in higher education. This paper proposes a control and interfacing software stack built on the Robot Operating System (ROS) and a simulation of a 3D printable 6 degree of freedom (DoF) robotic arm. Both the physical and simulated robot will be controllable through various inputs — a game controller, application programming interface (API), or terminal commands — and be able to perform specific tasks autonomously. The functionality of the final project will be demonstrated through an example of The Towers of Hanoi, control through an external API, and MATLAB control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots can be used to teach a wide number of classes and concepts including basic programming, concurrent programming, dynamics and control, mechatronics, engineering, electronics, forward and inverse kinematics, computer vision, and path planning [3, 4, 5, 6]. Currently there are few to no standardized robotic arm platforms with ready-made software stacks able to use both physical and simulated robots for teaching in higher education.

This platform will allow students to implement and test additions and modifications to the software on their own computers through the simulation environment, then come in and test their code on the physical robot, without any need for translation or modification of their code to make it work in the real world. Using practical examples has been shown to be beneficial to learning, and especially helps to interest students in the subject and aid in the self-learning process.

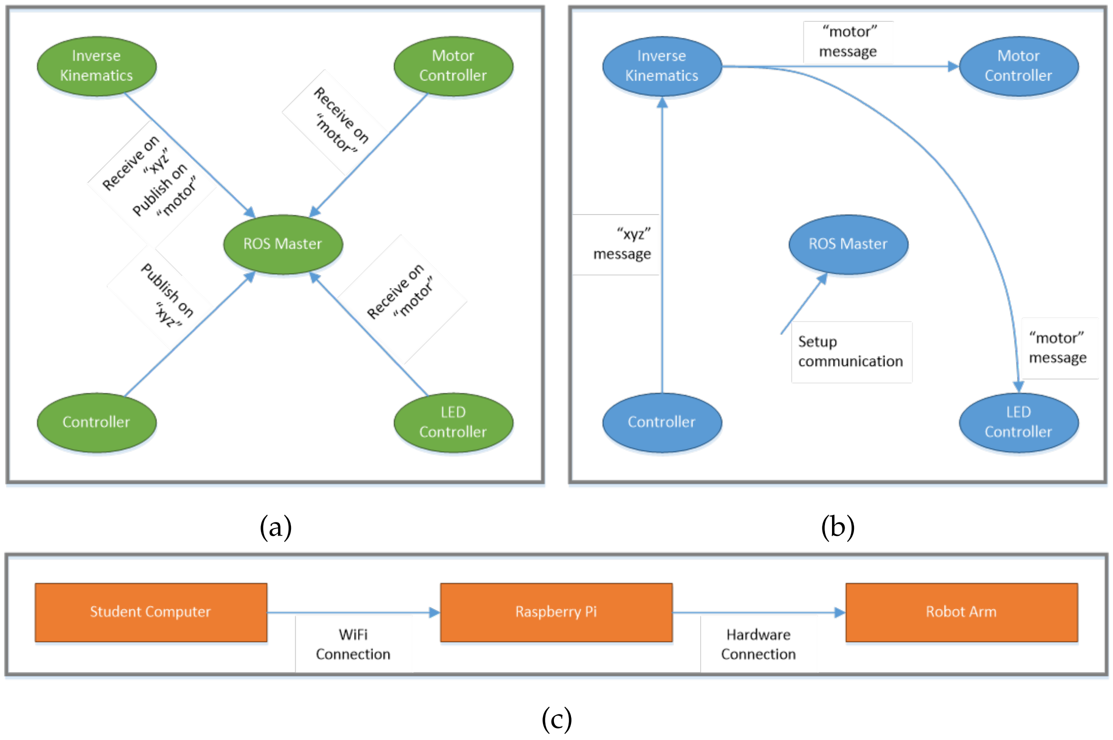

Figure 1.1: (a) When the ROS system first starts up each node connects to the ROS Master and gives it messages the node publishes and receives on. (b) Each receiving node will then connect directly to the sending nodes for message passing. (c) The system will be implemented on a Raspberry Pi and take input from some source to control the robot arm [1].

1.1 Goals and Requirements

The software control stack for the project will be built on the Robot Operating System (ROS). ROS has become a popular robotics platform for use in many types of robots [1]. The ROS platform is detailed in Figure 1.1. The system consists of two parts: the first part is the nodes of the system shown in Figures 1.1a and 1.1b. Nodes are the programs of the system and implement hardware integration, services, and interaction with the outside. Nodes may be custom built for the application or may come from the ROS community allowing hardware integration and other non-focus items to be done faster. The second part of the system is message passing, and each node is able to publish and receive messages. ROS controls the message passing between nodes and sets up the interaction between nodes as shown in Figure 1.1a. After the initial setup of the communication, messages are passed directly from one node to the next, as shown by the blue graph in Figure 1.1b.

The system of nodes and messages makes any software stack created on ROS a distributed system allowing for implementation flexibility. The functionality of the project will be implemented in the control of both a 3D simulation of and open-source 3D printed 6 DoF robotic arm by Andreas Hölldorfer [7]. Students can implement and reimplemented a function of the system by adding or recreating a node, leaving the rest of the system fully functional.

The functionality and movement of the robot will be shown through pre-programmed tasks such as performing the Towers of Hanoi. It will be able to perform real-time movement based on inputs from a game controller, API, and command line. It will also be able to be used in other more software focused classes through the API provided or through MATLAB integration.

1.2 Motivation

This project will allow science, technology, math, and engineering (STEM) students to work on a simulated and physical robot in class. The final product will make robotics learning and research easier by providing a robot and a software stack ready-made for immediate use and modification as opposed to each university and class starting from scratch. This allows the class time and projects to focus on learning objectives rather than the nuts and bolts of the specific system.

Chapter 2

Background

2.1 Education

Robots can be used to great effect in the area of education, and recently a greater emphasis has been placed on the subject due to a focus on science, technology, engineering, and math (STEM) especially in lower education [3]. Proposals of programs to introduce robotics into classrooms are very common. Unfortunately most use solely a simulated environment, or a physical robot, but few use both. A purely simulated approach can be dissatisfying to students, as seeing a finished final physical product provides a great deal of motivation and tangibility to the experience making classes engaging and attractive [8, 3]. A purely physical system can also be at a disadvantage, as students may only be able to work on the robot while they are physically in the lab. A physical teaching tool can have a large effect on the learning process, and if used along side virtual systems they both can provide a lab environment which engages students in the learning process [9, 10, 5, 11].

Robotic arms specifically have been used to teach in the areas of basic programming, concurrent programming, dynamics and control, mechatronics, engineering, electronics, forward and inverse kinematics, computer vision, and path planning [3, 4, 5, 6].

Many examples of robots designed for use in a classroom environment have been proposed for use in lower education teaching. Some of the most basic robots have been used in the area of early education[11, 12, 13]. These robots many times use graphical programming languages as is the case of the AERobot designed by Rubenstein et. all [11]. This robot was designed to be complex enough to do interesting tasks, such as obstacle detection and line following, but cheap enough to be affordable in mass to any community [11]. Another example of robots designed for lower education are the LEGO NXT robots used in the FIRST LEGO League competitions [12, 13]. These robots are used in competitions for kids from elementary to high-school, and allows them to design and program the robots to accomplish tasks in the competition [12, 13]. Though these robots are useful for teaching entry level programming and for introducing the concepts of robotic programming, they do not focus as much on the algorithms of the robot or how the robot actually functions, only the higher level functionality.

The second area for use of robots is in universities, colleges, and other forms of higher education. This demographic usually focuses more on the intricacies of robot construction, control algorithms, such as forward and inverse kinematics, and sensor processing. CHARM is a robotics course developed just for this purpose by Singh et. all [4]. This course teaches students how to design and program a robot for coin sorting, and in the process teaches kinematics, path planning, and robot vision through a coin sorting project [4]. Though this robot is good for teaching a number of concepts it only uses a three degree of freedom (DoF) SCARA robot arm. This makes the robot good for teaching in two dimensions, but does not allow for three dimensions. The robot may also be hindered in a lab environment because it does not propose a framework to build on requiring students to write the entire control system every time rather than focusing on the algorithms of the system one at a time, therefore a bottom up approach must be used, and the robot is only useful for the one class, not for a range of classes.

2.2 3D Printing

3D printing has quickly become a common method of rapid prototyping in recent years [14]. This is due to advances in 3D printing technologies and standardizations making more wide spread use possible [14]. 3D printers are now common in universities for use in research and product design. This makes the method ideal for use in producing products in a distributed fashion for education. 3D Printing has already been used for robotics research in universities as is the case in the 3D printed hand by Mukhtar et. all [15]. Poppy is another 3D printed platform designed for education by Lampeyre et. all [16]. Poppy has been used in several research projects at various universities since its design [17, 18].

Several robotic arm designs are available for free from various sources and makers. There are many simple robot designs available, but these designs have severely limited degrees of freedom [19, 20, 21]. There are several robot designs with more degrees which would be better suited for a general purpose robots. One such robot is the five axis Thor robot by AngelLM, unfortunately a five axis robot may not be sufficient enough for more advanced classes [22]. Another advanced design is the Zortrax robot, this robot looks good and is a simple sleek design, unfortunately not all of the joints are motorized making it unusable for control classes [23]. The Moveo arm by BNC3D is a good viable option as a six axis fully designed robot arm, this would make a good robot for use in a class environment [24]. The last 3d printed robot design is by Andreas Hölldorfer [7].

2.3 Simulation and Control Software

Simulation software is very common for a variety of different robots. Several popular robotics simulators exist [25]. These systems include the Virtual Robotics Toolkit, Robot Virtual Worlds, RoboDK, Microsoft Robotics Developer Studio, Webots, and Gazebo [25].

Both the Virtual Robotics Tool Kit and Robot Virtual Worlds simulators are designed for use with LEGO Mindstorms, Vex Robots, and other LEGO based systems, and primarily target early STEM education [25]. The systems are not as flexible as others due to their limited target audience, and though they have been used in higher education, the platform is aimed at first time programmers [26, 12, 27, 28, 29, 30]. The limits of the system force any modification to the programming and control of the it to be done in round about methods with external controllers [29]. The RoboDK, this software is designed for use in industrial robots and does not allow for externally controlled or open platform robots [25]. Microsoft Robotics Developer Studio has support for a wide array of platforms and is supported by a large company [25]. Though this has been a well received platform, support for the software has been canceled, meaning that any knowledge gained using the platform will be useless as the students go on to future projects [25]. Webots and Gazebo both officially support the ROS platform, and can be used to simulate any robot design [25]. Webots is a closed source paid platform, and therefore the system cannot be as modified for specialized uses [25, 31]. Gazebo, in contrast, is a free, open-source platform and has been used in developing interactive robots extensively [25]. Gazebo is also can be compiled and run on Linux, OSX, or Windows [32]. The final simulator available is RViz [33]. RViz is packaged with ROS in the full desktop installation [33]. RViz works only on Ubuntu Linux, and does not have support for other operating systems [33].

There are several different software stacks available for robot control. Many of the software stacks are custom and designed for a single robot. This is the case for systems such as BRACON by Rivas et. all [34]. If the software is not widespread enough as an open source software, or is proprietary, it may not allow for the best use in education as it does not allow for modification for new systems. Many of these systems only allow limited methods of interaction such as through gCode which is simply a communication format, and does not allow dynamic movement or feedback.

The Robot Operating System (ROS) is a promising new open source robotic framework for use in a wide array of robots. ROS has been used in previous educational robotics projects such as Nelson [35]. Other projects have focused on extending ROS for use in MATLAB such as in the case of ros4mat [36]. ROS comes with both control software and a simulation environment which can be controlled simultaneously. The software is modular, as shown if Figure 1.1 and can be distributed across devices [1].

Chapter 3

Proposal

Practical tools and examples can be used to great effect in a classroom environment. This paper will propose the use of an open source 3D printed arm by Andreas Hölldorfer, software and hardware control platform built on ROS, and simulation environment for use in a the classroom for the teaching of robotics and control at Southern Adventist University.

3.1 Hardware Requirements

This section will outline the hardware required for the project.

3.1.1 Robot

As seen in Figures 3.1 and 3.2, the project will use the 3D printed robotic arm created by Andreas Hölldorfer [2]. It is currently one of the most advanced open source robotic arms available. The robotic arm is cheap to produce making it ideal for a higher education.

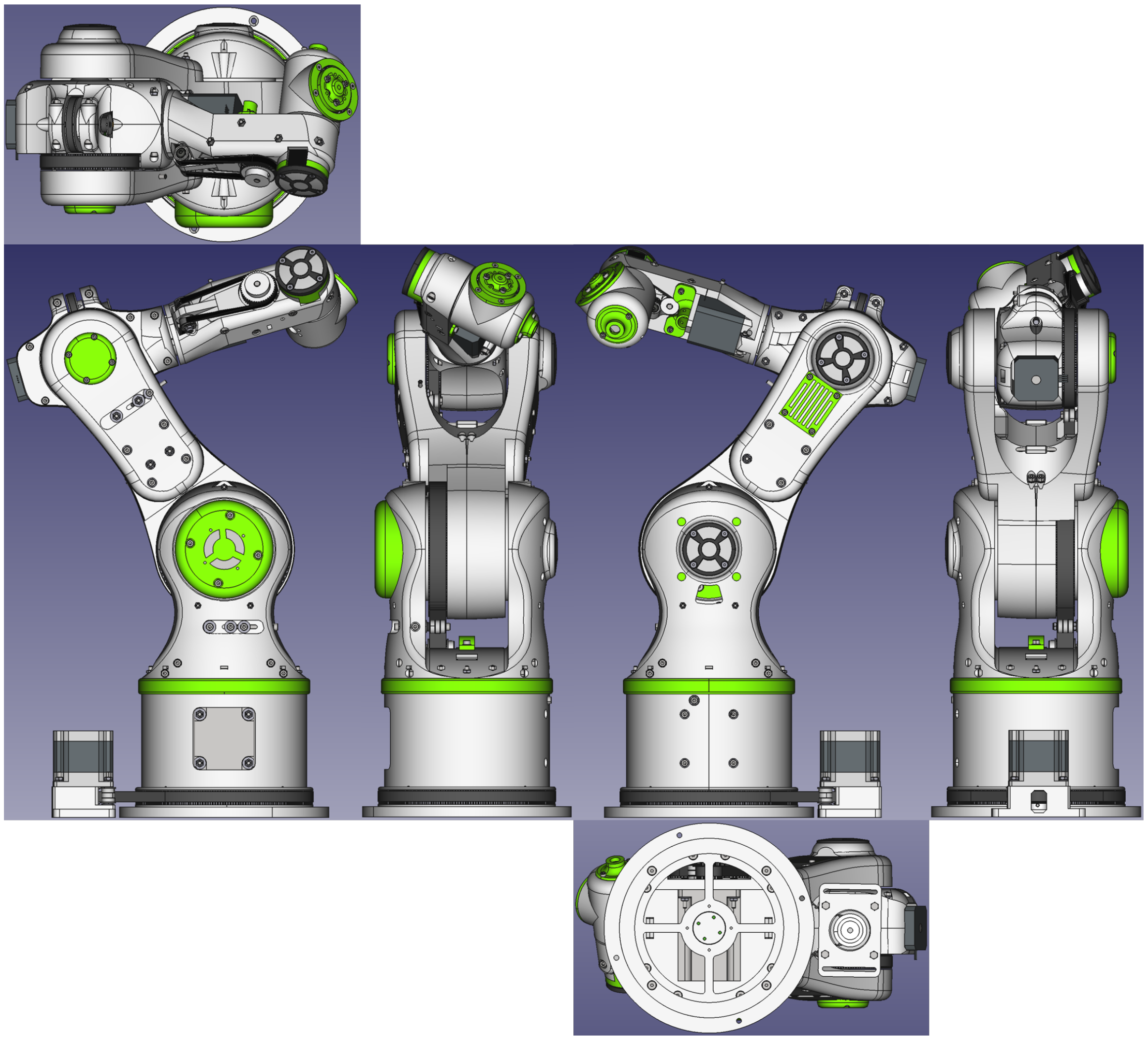

Figure 3.1: Orthographic views of the 6-degree of freedom (dof) arm designed by Andreas Hölldorfer. top view (top), left side view (far left), front view (center left), right side view (center right), back view (far right), and bottom view (bottom) [2].

3.1.2 Electronics

The controller for the robot will be primarily comprised of a Raspberry Pi computer running either the Raspian Operating System (OS). The motors will be controlled through off-the-shelf stepper motor controllers with absolute position rotary encoders to tell the angle of each joint. All of the motors and hardware on the robot will be controlled through a microcontroller connected to the Raspberry Pi. This

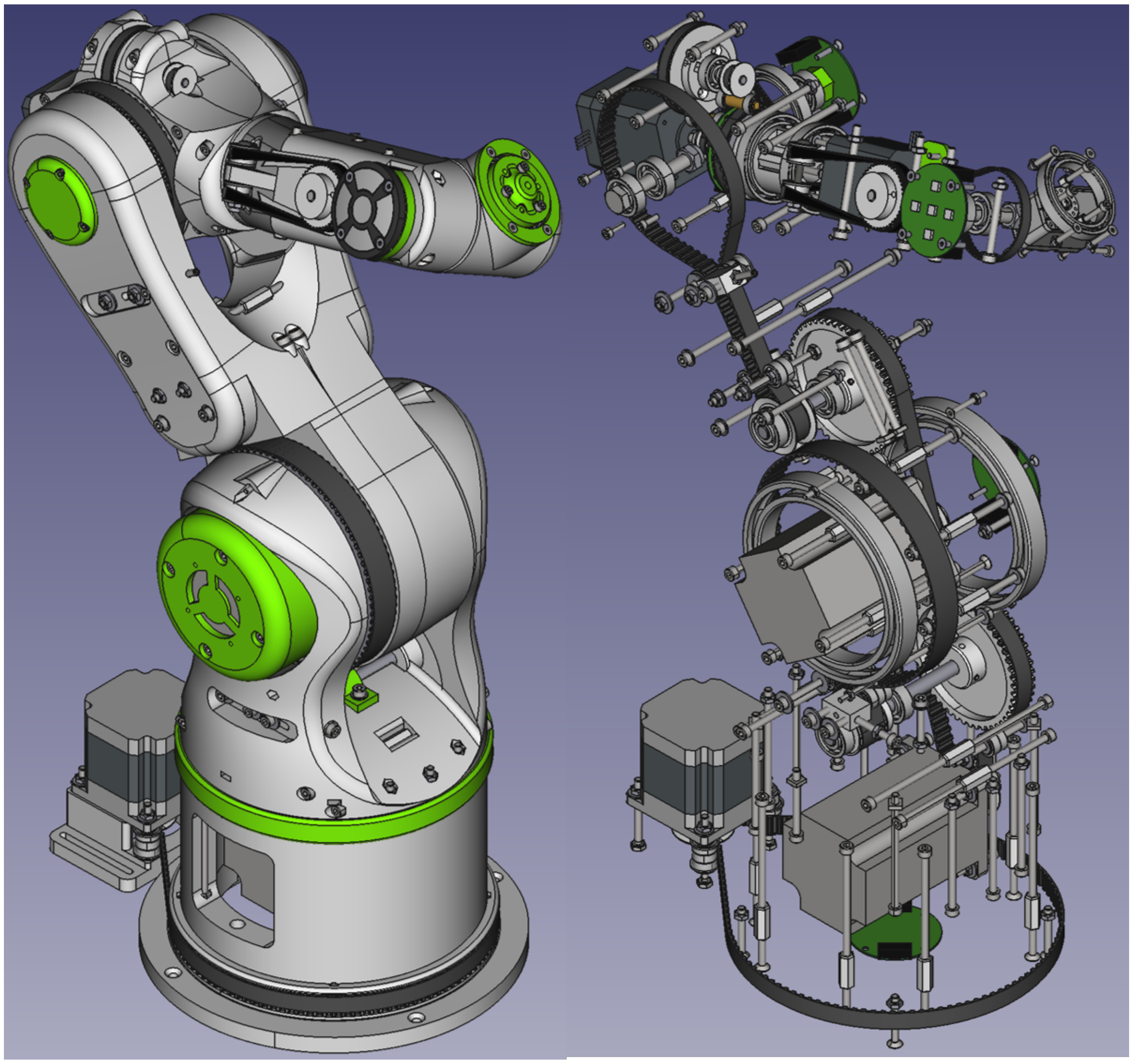

Figure 3.2: Isometric views of the 6-degree of freedom (dof) arm designed by Andreas Hölldorfer. Complete arm (left), arm with shell removed (right) [2].

will allow for real-time control and an an application programming interface (API) abstraction to simplify hardware control.

3.1.3 Interaction

The system will be able to be accessed through either an Ethernet or Wi-Fi connection to a computer, or a Bluetooth connection to a Wii like controller. Ports will also be made available on the base to allow an HDMI cable, keyboard, and mouse to be plugged into the Raspberry Pi directly.

3.2 Software Requirements

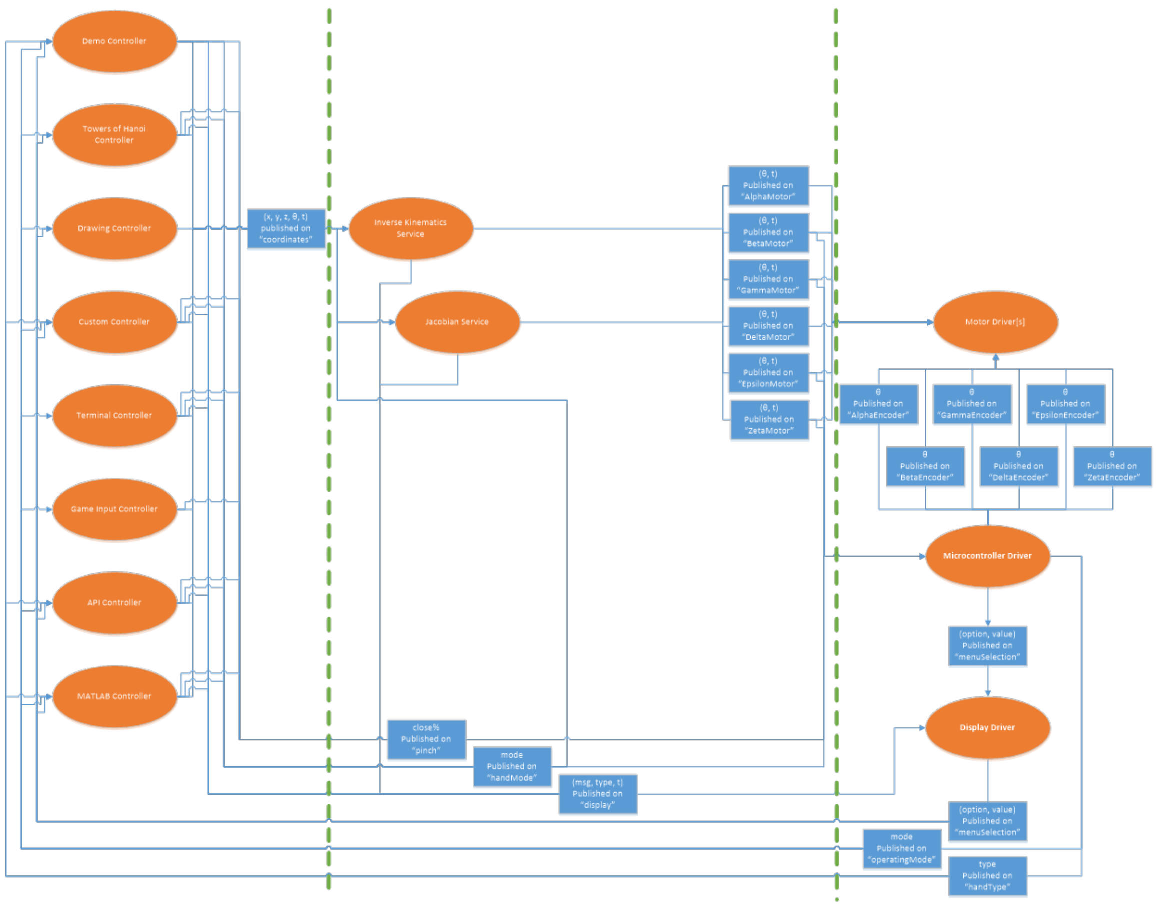
Demo Controller
Towers of Hanoi Controller
Drawing Controller
Custom Controller
Terminal Controller
Game Input Controller
API Controller
MATLAB Controller
(x, y, z, θ, t) published on "coordinates"
Inverse Kinematics Service
Jacobian Service
(θ, t) Published on "AlphaMotor"
(θ, t) Published on "BetaMotor"
(θ, t) Published on "GammaMotor"
(θ, t) Published on "DeltaMotor"
(θ, t) Published on "EpsilonMotor"
(θ, t) Published on "ZetaMotor"
Motor Driver[s]
θ Published on "AlphaEncoder"
θ Published on "GammaEncoder"
θ Published on "EpsilonEncoder"
θ Published on "BetaEncoder"
θ Published on "DeltaEncoder"
θ Published on "ZetaEncoder"
Microcontroller Driver
(option, value) Published on "menuSelection"
Display Driver
close% Published on "pinch"
mode Published on "handMode"
(msg, type, t) Published on "display"
(option, value) Published on "menuSelection"
mode Published on "operatingMode"
type Published on "handType"

Figure 3.3: Graph of the ROS nodes (orange) and messages (blue), showing the publishers and subscribers used for the project.

The software for the robotic arm will be built on ROS. This will allow for a modular design for use in educational sections. The use of ROS also means the framework will be maintained by a large community of contributers and reduce the maintenance requirements of the project in the future. ROS also provides a large number of debugging, visualization, and development tools requiring fewer pieces of the system to be custom made.

A ROS system is made up of nodes, shown in orange in 3.3 [1]. Each node may publish and subscribe to topics to communicate between nodes shown in blue on Figure 3.3 [1]. Some control nodes may also bypass others to implement lower functionality, this is shown in yellow on Figure 3.3 [1]. For the robot arm project the control system is made of several specific parts controllers, shown in the first column of nodes, services, shown in the second, and drivers, shown in the third. The following section will detail the various nodes of the system shown in Figure 3.3.

3.2.1 Controllers

Controllers of the system serve as primary inputs to the arm. These nodes all will publish an x, y, and z end-effector coordinates, an end-effector angle (θ), and a time (t) parameter on the ″inCoordinate″ message. A few of the nodes will publish the hand messages of close percentage and hand mode on ″pinch″ and ″handMode″ respectively. These will tell the hand how to pinch and grab things in different configurations. Some controllers may also publish to a character display for informational messages. These messages will contain a message (msg), with an information string, a type, showing whether the message is showing some information, a warning, or an error, and a time (t), to tell how long to display the message. The control nodes may also take several inputs. The mode message published on ″operatingMode″ will tell the controllers which one is currently supposed to be controlling the system. The type message published on ″handType″ and will tell the controllers which hand is currently being used.

3.2.1.1 Demo

The Demo Controller will continuously show off the capabilities of the arm for use in demonstration for introductions and recruitment purposes. This mode will not take in any input from the outside world. This mode will be assigned number 0 in the mode options list.

3.2.1.2 Towers of Hanoi

The Towers of Hanoi Controller will direct the robot to perform an optimal Towers of Hanoi problem. No outside input is needed to perform the task.

3.2.1.3 Drawing

The Drawing Controller will draw the southern and school of computing logos on a white board. When the drawing is done the arm will erase the board and start again.

3.2.1.4 Custom

The Custom Controller will allow for students to make a control node for class work and allow for interaction within the systems framework. Outside input may be taken in from any source necessary for the node to function, including other nodes, external controllers, MATLAB, or web based inputs. It may also bypass any node in the system by publishing their output messages directly, or receive from any node in the system by accepting their messages, this is not shown in Figure 3.3 since it is dependent on the implementation being done.

3.2.1.5 Terminal

The Terminal Controller will allow for input from the Linux terminal. It will be able to take in x, y, and z end-effector coordinates, an end-effector angle (θ), and a time (t) parameter and send them to the arm. Alternately a user may also send the robot a series of motor angles and times to move the robot to a position. This functionality will bypass the service modules and send messages to the motors directly, this is not shown in the graph.

3.2.1.6 Game Input

The Game Input Controller will allow for input from a Wiimote through a bluetooth connection with the Raspberry Pi. Open source drivers are available for the Wiimote through XWiiMote [37]. The application integration is also provided for the Python and Perl languages [37]. The node will take the accelerometer and gyroscope inputs from the Wiimote and use them to control the position and orientation of the end-effector.

3.2.1.7 API

The API Controller will allow for integration with remote programs through an Ethernet or WiFi connection. The API will allow for x, y, z end-effector coordinates, an end-effector angle (θ), and a time (t) parameter. The input values will then be sent to the rest of the system.

3.2.1.8 MATLAB

The MATLAB Controller will allow for integration between MATLAB Simulink and the robot. This will allow for programs to be written in MATLAB and executed through the physical robot This module, unlike the other control modules in the system, will be able to publish on any message in the system bypassing other modules. This setup is not shown in Figure 3.3, as it connects to all other modules.

3.2.2 Translators

Translators in the system act as data modifiers and middle-men between the controllers and drivers of the system. multiple translators may be connected in series or parallel to modify data in multiple steps.

3.2.2.1 Inverse Kinematics

The Inverse Kinematics (IK) service subscribes to ″coordinates″ messages, and publishes ″Alpha″ through ″Zeta″ ″motor″ messages. The IK service will take the end-effector input coordinates and angle, and movement time parameters convert them into motor angles for the system.

3.2.2.2 Jacobian

The Jacobian service subscribes to ″coordinates″ messages, and publishes ″Alpha″ through ″Zeta″ ″motor″ messages. This service will strait lines between the current and given points.

3.2.3 Drivers

The drivers of the system form the final end-points. They provide the integration between the controllers and the physical hardware.

3.2.3.1 Motors

For each hardware stepper motor controller there is a motor driver. These controllers will tell the motors what angle to move to, and control the time it takes to get from the initial to final position. The motors will receive angle and time information on the ″Alpha″ through ″Zeta″ ″Motor″ messages. current angle information will be recieved on the ″Alpha″ through ″Zeta″ ″Encoder″ messages. This information will be used to initialize the arm from any position.

3.2.3.2 Microcontroller

The Microcontroller Driver will interpret and relay all of the messages between the microcontroller components and the other nodes. This node may be split out into multiple nodes in the implementation with a single node interfacing with the microcontroller. The first portion of the microcontroller driver is the encoder LEDs These LEDs will show the status of the motors and if any moves are invalid for a motor. This information will be received on the ″Beta″, ″Gamma″, ″Epsilon″, and ″Zeta″ ″Motor″ messages. The LEDs will receive on the ″display″ message and use the type information to display errors on all of the encoders. The encoder drivers will publish angle information on the ″Alpha″ through ″Zeta″ ″Encoder″ messages.

Menu control buttons will also be controlled through the Microcontroller Driver. The buttons will publish on ″menuButtons″ with the number of the button which has been pressed. The last component is the hand which will control the servos of the hand and detect which hand has been plugged in. The hand will receive pinch values on the ″pinch″ message, and hand mode information on the ″handMode″ message. Information about the hand currently plugged in will be relayed on the

″handType″ message.

3.2.3.3 Display

The display driver will control the RGB back-lit display on the base of the robot. The display will show menu items and informational messages for and from the modules. The color of the display will be determined by the type of message incoming and the mode of the arm. The display will publish option and value pairs on the ″menuSelection″, and will receive message, type, and time information for display on the ″display″ message, and menu button presses on ″menuButtons″.

3.3 Educational Uses

A robotic arm can be used to teach many difficult concepts in robotics and other classes as shown in previously mentioned works. The initial use of this particular arm will be specific to a graduate robotics class, though future uses could go far beyond just one class. The initial concepts this robot is intended to teach include: robot construction and design, kinematics, inverse kinematics, Jacobians, and robot control frameworks. This list may be expanded later on to include more concept and areas in the future.

3.3.1 Controller Creation

Controller creation will be used for teaching robot design and movement, this may require bypassing modules, and robot control frameworks. The creation of new controller modules will provide an introduction to ROS programming. This piece may also be used later on to teach higher level concepts in other areas and possibly even other classes, as it should be simple requiring a very shallow learning curve to accomplish.

3.3.2 Module Creation

Module creation will be used for the teaching inverse kinematics, Jacobians, and robot control frameworks. This will be accomplished by removing the node which accomplishes the intended function to be taught, and having the students recreate the module. All other pieces of the system will still be running, allowing the new module to tested without causing the student to create the rest of the system from scratch. This will allow these pieces to be taught using more concrete, hands-on experiences, without taking the time to recreate an entire system.

3.3.3 MATLAB Integration

MATLAB integration can be used for the teaching of many concepts quickly through writing MATLAB scripts. This path also does not require any learning of ROS, if it is preferred for the class or module. The MATLAB Integration may also be used to bypass the service modules in order to teach their functionality through MATLAB, instead of compiled code, the bypassing functionality is not shown in Figure 1.1.

3.3.4 Terminal Control

Terminal control will allow for initial teaching and testing of the robot in class. It will allow students to either enter x, y, z, and θ coordinates, or allow the entry of motor angles. This will allow students to experiment with the robot in order to see how the system reacts and functions to various inputs.

3.3.5 API Interaction

API interaction will allow for remote control of the robot for students without ROS installed. This functionality can also be used for the teaching of future robotics concepts, as well as concepts in other classes which are less concerned with how the robot is controlled and are more concerned with higher level functionality, such as artificial intelligence (AI), robot vision, or algorithms.

3.4 simulation

For testing the software, along with the physical robot, a simulator will be used to show the output from the software. The project will utilize the Gazebo simulator. This simulator works on the three major operating systems, allowing it to run outside of a VM making it more accessible. This simulator is free and open source.

3.5 Bill of Materials

Category	Total	Percentage
Interaction	$68.21	5%
Power	$108.37	7%
Control	$140.01	10%
Mechanics	$678.04	46%
Plastic	$101.95	7%
Motors	$170.98	12%
Encoders	$100.38	7%
Miscellaneous	$100.00	7%
Total	$1467.94	

Table 3.1: The list of general costs are outlined in the above table. For a full list of materials see table B.1

3.6 Tasks and Milestones

Milestones	Task Group	Hours
1	Documentation	40
1	Research	260
2	ROS Setup	4
2	Raspberry Pi Setup	18
3	Boot Sequence	14
4	Simulator	20
5	Mechanics Construction	44
5	Electronics Construction	18
5	Circuit Design	30
6	Microcontroller Programming	72
7	Motor Drivers	14
7	Encoder Drivers	10
7	Encoder LED Drivers	10
8	Hand Drivers	10
9	Menu Display Driver	18
9	Display Driver	10
10	Custom Controller	10
11	Inverse Kinematics Service	18
11	Jacobian Service	18
12	Demo Controller	14
12	Towers of Hanoi Controller	18
12	Drawing Controller	18
13	Terminal Controller	12
13	Game Input Controller	20
13	API Controller	22
13	MATLAB Controller	22
Total Hours		764

Table 3.2: The above table shows the list of task groups for each module. The modules and tasks are in the order of completion for the project. A full list of tasks can be found in table A.1 in appendix section A.1

The table A.1 shows the list of milestones shown as project modules in the first column. Under each milestone is a list of task groups and under them a list of tasks. The hours for each milestone, group, and task is shown in the last column.

Most of the milestones must be completed in the order listed. The Bill of Materials and ROS Research in Module 1 has already been completed.

The deliverables for the project will include the software stack, functional 3D printed robotic arm, system documentation and simulation of the robot arm. The system will be set up for use in a classroom environment ready for labs.

Chapter 4

Testing/Evaluation Plan

Testing of the robot will include hardware, software, and acceptance testing phases. The hardware phase will test the precision, accuracy, and maximum capacity of the arm. The software phase will test the correctness of the program. The acceptance testing phase of the project will determine when the project has been completed.

4.1 Hardware Testing

Precision will be tested by attaching a needle to the end of the arm and marking pre-determined places on a piece of paper, 10 times for each place. The largest delta from the pre-determined point will give the precision. The precision of the robot should be within 0.2mm.

Accuracy will be tested by attaching a needle to the end of the arm and marking the same point on a piece of paper 10 times. The largest difference between any two points is the accuracy of the robot. The accuracy of the robot should be within 0.04mm.

The arm should be able to lift at least 2kg of weight fully extended. This has been determined by the testing of Andreas Hölldorfer.

4.2 Software Testing

Software testing will be done through unit testing of each node. The unit testing will use rostest for connecting to and testing the ros nodes, gtest for testing the nodes written in c++, and unittest for testing the nodes written in python [38].

4.3 acceptance Testing

The project will be deemed to be completed when all of the nodes function as outlined in the software requirements testing, and when the software and hardware tests are completed and passed.

Chapter 5

Conclusion

The proposed project defines a standard robotic platform with a ready-made software stack able to use both a physical and simulated robot for teaching in higher education. The project will be implemented on the ROS platform using the RViz simulator. The open-source 3D printable 6-dof arm by Andreas Hölldorfer will be used for the physical robot with all of the control electronics designed in-house. Testing of the system will be done through unit testing via rostest, gtest, and unittest, and physical testing will test the precision, accuracy, and load capacity of the arm. The system will be demonstrated through the running of the control nodes of the software stack.

Appendix A

Requirements

A.1 Milestones and Tasks

Milestones	**Group**	**Task**	**Hours**
Milestone 1			**100**
	Documentation		40
	Research		260
		Research	40
		BOM Writing	60
		Proposal Paper	120
		Presentation Work	40
Milestone 2			**22**
	ROS Setup		4
		Running ROS	2
		Initial Project Setup	2
	Raspberry Pi Setup		18

Raspian Setup	2′
WiFi Setup	8
Program Setup	4
Raspian Imaging	2
Login Setup	2
Milestone 3	**14**
Boot Sequence	14
Display Programming	2
Encoder Programming	4
Encoder Initialization	2
Raspberry Pi Startup Tasks	2
ROS Launch List	4
Milestone 4	**30**
Simulator	20
Simulator Setup and Modeling	14
Robot Control	6
Milestone 5	**92**
Mechanics Construction	44
3D Printing	16
Robot Assembly	16
Base Construction	6
Base Design	6
Electronics Construction	18
Assembly	16

Testing	2
Circuit Design	30
Encoder Board Design	4
Encoder Board Milling	2
Encoder Board Construction	4
Regulator Board Design	6
Regulator Board Milling	2
Regulator Board Construction	2
Display Board Design	6
Display Board Milling	2
Display Board Construction	2
Milestone 6	**72**
Microcontroller Programming	54
Setup	8
Motor Control	6
USB Communication	16
LED Control	6
Encoder Reading	8
Display Control	12
Display LED Control	8
Hand Control	8
Milestone 7	**22**
Motor Drivers	12
Node Writing	8

Unit Testing	4
Encoder Drivers	8
Node Writing	4
Unit Testing	4
Encoder LED Drivers	8
Node Writing	4
Unit Testing	4
Milestone 8	**6**
Hand Drivers	8
Node Writing	4
Unit Testing	4
Milestone 9	**20**
Menu Display Driver	16
Node Writing	12
Unit Testing	4
Display Driver	8
LED Control	2
Display Control	2
Unit Testing	4
Milestone 10	**6**
Custom Controller	8
Template/Node Writing	4
Unit Testing	4
Milestone 11	**28**

Inverse Kinematics Service	16
Node Writing	12
Unit Testing	4
Jacobian Service	16
Node Writing	12
Unit Testing	4
Milestone 12	**38**
Demo Controller	12
Node Writing	8
Unit Testing	4
Towers of Hanoi Controller	16
Node Writing	12
Unit Testing	4
Drawing Controller	16
Node Writing	12
Unit Testing	4
Milestone 13	**60**
Terminal Controller	10
Node Writing	6
Unit Testing	4
Game Input Controller	18
Node Writing	10
Controller Integration	4
Unit Testing	4

API Controller	20
Node Writing	16
Unit Testing	4
MATLAB Controller	20
Node Writing	16
Unit Testing	4
Total Hours	732

A.2 Hardware

1. The 3D printed open-source robotic arm by Andreas Hölldorfer will be built and used for the physical robot [7].
2. A bill of materials will be created for all of the parts and equipment necessary for the building of the robot.
3. A base will be designed to hold the control components of the robot.
4. several circuit boards will be designed for the robot for power conditioning and regulating, microcontroller and peripheral interfacing, and encoder boards.
5. The robot will include a Raspberry Pi for control with a microcontroller co-processor for real-time tasks.

A.3 Software

A.3.1 Microcontroller Firmware

1. The microcontroller should initialize all of the components on the arm, then make them show a of the arms functionality while the Raspberry Pi boots up.

2. The microcontroller should control the motors, display, LEDs, encoders, and servos. This is to allow an abstraction layer from the hardware and real-time functionality.

3. The microcontroller should communicate with the Raspberry Pi over either a USB or GPIO.

A.3.2 Raspberry Pi Software

1. The Raspberry Pi should run the Raspian Operating System.

2. The Raspberrry Pi will run ROS for the robot control.

3. The ROS system should contain a node for controlling the motors.

4. The ROS system should contain a node for controlling the encoders.

5. The ROS system should contain a node for reading the encoder LEDs.

6. The ROS system should contain a node for controlling the hand servos.

7. The ROS system should contain a node for controlling the display text and managing the menu system.

8. The ROS system should contain a node for controlling the display LEDs.

9. The ROS system should contain an inverse kinematics service node.
10. The ROS system should contain a jacobian service node.
11. The ROS system should contain a custom controller node.
12. The ROS system should contain a demo controller node.
13. The ROS system should contain a Towers of Hanoi Controller node.
14. The ROS system should contain a drawing controller node.
15. The ROS system should contain a terminal controller node.
16. The ROS system should contain a game input controller node.
17. The ROS system should contain an API controller node.
18. The ROS system should contain a MATLAB controller node.

A.3.3 Simulation and Remote Control Software

1. The remote computer should run the simulator for the robot.
2. The remote computer should be able to run all of the Raspberry Pi nodes for use with the simulator or remote operation.
3. The remote computer should be able to run any of the control nodes for remote programming and testing of the system.
4. The remote system should be able to run MATLAB to integrate with the MATLAB node of the robot.
5. The remote system should be able to run a web browser to interact with the API node of the robot.

6. Remote control of the system should be possible through a Wii like Bluetooth connected controller.

A.4 Installation

1. The user should be able to install the robot software by copying the project files from the repository to the project on their own system.

A.5 Running

1. The project should be able to be run through a ROS launch file which will launch all of the nodes on the system.
2. The system on the Raspberry Pi should startup automatically once Raspian has started as a background task.

A.6 Modification

1. Students should be able to modify the system by removing and rewriting a node.
2. Students should be able to write the code for the custom controller to modify the arm for other uses.
3. the physical robot should be able to be upgraded by printing new parts for it as the open-source project progresses.

Appendix B

Bill of Materials

#	Quantity	Item	Cost/Item	Total Cost
Interaction				
1	1	RGB LCD 20x4	$24.95	$24.95
2	1	4 Switch Keypad	$21.07	$21.07
3	1	ATMEGA32U2-AU-ND	$2.99	$2.99
4	3	Power Transistor	$3.49	$10.47
5	1	USB Type A Connector	$0.40	$0.40
6	1	Micro USB Connector	$0.46	$0.46
7	1	Panel Mount HDMI	$5.95	$5.95
8	1	Panel Mount Ethernet	$4.95	$4.95
9	1	Panel Mount USB A	$3.95	$3.95
Power				
13	1	Power Cord	$3.81	$3.81
14	1	Illuminated Switch	$5.98	$5.98
10	1	AC/DC Converter	$92.95	$92.95

11	1	J100 Female Connector	$0.19	$0.19
12	1	J300 Female Connector	$0.34	$0.34
15	2	Micro USB Cable	$2.55	$5.10
Control				
16	1	Raspberry Pi 3	$39.95	$39.95
17	3	Stepper Motor Driver 10 - 32 VDC	$20.70	$62.10
18	2	Stepper Motor Driver 12 - 45 VDC	$18.98	$37.96
Mechanics				
19	19	624 Ball Bearing	$4.75	$90.25
20	11	608 Ball Bearing	$1.49	$16.39
21	4	61807 Ball Bearing	$7.20	$28.80
22	2	61818 Ball Bearing	$78.45	$156.90
23	4	626 Ball Bearing	$4.14	$16.56
24	4	696 Ball Bearing	$4.30	$17.2
25	11	F624 ZZ Ball Bearing	$7.89	$86.79
26	3	DIN 912 - M4 x 20mm	$0.66	$1.98
27	5	DIN 912 - M4 x 25mm	$0.09	$0.45
28	26	DIN 912 - M4 x 30mm	$0.10	$2.60
29	7	DIN 912 - M4 x 35mm	$0.18	$1.26
30	3	DIN 912 - M4 x 40mm	$0.21	$0.63
31	10	DIN 912 - M4 x 50mm	$0.17	$1.70
32	19	DIN 912 - M4 x 55mm	$0.12	$2.28
33	5	DIN 912 - M4 x 60mm	$0.22	$1.10
34	18	DIN 912 - M4 x 70mm	$0.30	$5.40

35	6	DIN 913 - M4 x 3mm	$0.05	$0.30
36	100	DIN 913 - M4 x 8mm	$0.05	$5.00
37	2	DIN 913 - M4 x 16mm	$0.05	$0.01
38	7	DIN 913 - M4 x 4mm	$0.04	$0.28
39	1	DIN 913 - M4 x 6mm	$0.04	$0.04
40	1	DIN 931 - M8 x 50mm	$0.39	$0.39
41	2	DIN 931 - M8 x 55mm	$0.42	$0.84
42	1	DIN 933 - M4 x 25mm	$0.10	$0.10
43	4	DIN 933 - M4 x 30mm	$0.12	$0.48
44	6	DIN 933 - M4 x 40mm	$0.16	$0.96
45	2	DIN 933 - M4 x 55mm	$0.16	$0.32
46	10	DIN 934 - M3	$0.04	$0.40
47	53	DIN 934 - M4	$0.04	$2.12
48	1	DIN 985 - M4	$0.04	$0.04
49	3	DIN 985 - M8	$0.10	$0.30
50	18	DIN 9021 - M4	$0.04	$0.72
51	12	DIN 912 - M3 x 10mm	$0.05	$0.60
52	7	DIN 912 - M3 x 16mm	$0.07	$0.49
53	6	DIN 912 - M3 x 20mm	$0.09	$0.54
54	4	DIN 125 - M3	$0.04	$0.16
55	68	DIN 125 - M4	$0.04	$2.72
56	4	DIN 125 - M5	$0.04	$0.16
57	1	DIN 125 - M6	$0.04	$0.04
58	5	DIN 125 - M8	$0.04	$0.20

59	12	DIN 7991 - M3 x 16mm	$0.04	$0.48
60	4	DIN 7991 - M3 x 20mm	$0.05	$0.20
61	6	DIN 7991 - M3 x 25mm	$0.07	$0.42
62	8	DIN 7991 - M4 x 25mm	$0.07	$0.56
63	8	DIN 7991 - M4 x 30mm	$0.09	$0.72
64	1	Timing Belt T5 - 500mm x 16mm	$14.69	$14.69
65	1	Timing Belt T5 - 510mm x 10mm	$9.78	$9.78
66	1	Timing Belt T5 - 630mm x 10mm	$11.55	$11.55
67	2	Timing Belt T5 - 340mm x 10mm	$7.91	$15.82
68	1	Timing Belt T2.5 - 317.5mm x 6mm	$6.70	$6.70
69	3	Timing Belt T2.5 - 200mm x 6mm	$6.11	$18.33
70	4	Synchronous Belt Pulley - T2.5 Z16 6mm	$4.76	$19.04
71	4	Synchronous Belt Pulley - T5 Z16 10mm	$5.76	$23.04
72	1	Synchronous Belt Pulley - T5 Z16 16mm	$6.05	$6.05
73	1	Synchronous Belt Pulley - T2.5 Z44 6mm	$5.77	$5.77
74	2	Synchronous Belt Pulley - T5 Z48 16mm	$16.71	$33.42
75	13	DIN 562 - M4	$0.04	$0.52
76	1	ISO 7380-1 - M4 x 30mm	$0.09	$0.09
77	1	ISO 7380-1 - M4 x 35mm	$0.10	$0.10
78	1	Shaft 8mm x 80mm	$1.18	$1.18
79	31	Spacer M4 20mm	$0.72	$22.32
80	4	Spacer M3 5mm	$0.35	$1.40
81	2	Tensioning Pin 3mm x 20mm	$0.08	$0.16
82	2	Tensioning Pin 3mm x 26mm	$0.08	$0.16

83	1	D Type Shaft - F6mm T15mm D6mm L50mm	$14.04	$14.04
84	1	Shaft - D80mm L120mm	$11.96	$11.96
85	2	Shaft - D7.5mm F5mm T5mm P4mm Q4mm	-	-
86	1	Shaft - D6mm L45mm T15mm Q5mm	$16.90	$16.90
Plastic				
87	6	PETG Printer filament White - 1kg	$18.99	$113.94
88	2	PETG Printer filament Red - 1kg	$25.99	$51.98
89	1	PETG Printer filament Natural - 1kg	$18.99	$18.99
Motors				
90	1	Nema 17 48mm Stepper Motor	$11.58	$11.58
91	1	Nema 17 60mm Stepper Motor	$10.72	$10.72
92	1	Nema 23 Stepper Motor	$19.14	$19.14
93	2	Nema 24 88mm Stepper Motor	$26.81	$53.62
94	1	HerculeX DRS-0101 Robot Servo	$60.34	$60.34
95	2	Analog Feedback Servo	$14.95	$14.95
Encoders				
96	2	APA102 2020 Dotstar LED	$5.95	$9.00
97	5	Magnetic Absolute Rotary Encoder	$8.68	$43.40
98	1	Circuit Boards	$35.00	$35.00
99	5	Encoder Magnet	$0.30	$1.50
100	10	Shrouded Header	$0.263	$2.63
101	10	Socket Connector	$0.301	$3.01
102	1	Ribbon Cable	$2.94	$2.94
Miscellaneous				

103	1	Other	$100.00	$100.00
Totals	**669**			**$1471.05**

Bibliography

[1] P. Bouchier. (2015, April) A gentle introduction to ros (and related technologies). [Online]. Available: https://dprgblog.files.wordpress.com/2015/04/2015marchrostalk-1.pdf (document), 1.1, 2.3, 3.2

[2] A. Hölldorfer. (2016, February) 3d printable robot arm. [Online]. Available: https://github.com/4ndreas/BetaBots-Robot-Arm-Project (document), 3.1.1, 3.1, 3.2

[3] M. F. Silva, B. Curto, and V. Moreno, "A robot in the classroom," in *Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality*, ser. TEEM '15. New York, NY, USA: ACM, 2015, pp. 197–201. [Online]. Available: http://doi.acm.org.ezproxy.southern.edu/10.1145/2808580.2808610 1, 2.1

[4] S. P. N. Singh, H. Kurniawati, K. S. Naveh, J. Song, and T. Zastrow, "Charm: A platform for algorithmic robotics education amp; research," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 2602–2607. 1, 2.1

[5] J. Shin, A. Rusakov, and B. Meyer, "Concurrent software engineering and robotics education," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15. Piscataway, NJ,

USA: IEEE Press, 2015, pp. 370–379. [Online]. Available: http://dl.acm.org.ezproxy.southern.edu/citation.cfm?id=2819009.2819068 1, 2.1

[6] F. Cuellar, D. Arroyo, E. Onchi, and C. Penaloza, "Irep: An interactive robotics education program for undergraduate students," in *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*, Oct 2013, pp. 153–158. 1, 2.1

[7] A. Hölldorfer. (2016, April) Chaozlabs. [Online]. Available: http://chaozlabs.blogspot.de/ 1.1, 2.2, 1

[8] T. L. Dunn and A. Wardhani, "A 3d robot simulation for education," in *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, ser. GRAPHITE '03. New York, NY, USA: ACM, 2003, pp. 277–278. [Online]. Available: http://doi.acm.org.ezproxy.southern.edu/10.1145/604471.604535 2.1

[9] T. Sapounidis, S. Demetriadis, and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools," *Personal Ubiquitous Comput.*, vol. 19, no. 1, pp. 225–237, Jan. 2015. [Online]. Available: http://dx.doi.org.ezproxy.southern.edu/10.1007/s00779-014-0774-3 2.1

[10] A. Saad and R. M. Kroutil, "Hands-on learning of programming concepts using robotics for middle and high school students," in *Proceedings of the 50th Annual Southeast Regional Conference*, ser. ACM-SE '12. New York, NY, USA: ACM, 2012, pp. 361–362. [Online]. Available: http://doi.acm.org.ezproxy.southern.edu/10.1145/2184512.2184605 2.1

[11] M. Rubenstein, B. Cimino, R. Nagpal, and J. Werfel, "Aerobot: An affordable one-robot-per-student system for early robotics education," in *2015 IEEE*

International Conference on Robotics and Automation (ICRA), May 2015, pp. 6107–6113. 2.1

[12] LEGO. (2016) Learn to program - it's easy. [Online]. Available: http://www.lego.com/en-us/mindstorms/learn-to-program 2.1, 2.3

[13] FIRST. What is first lego league. [Online]. Available: http://www.firstlegoleague.org/about-fll 2.1

[14] W.-P. Xu, W. Li, and L.-G. Liu, "Skeleton-sectional structural analysis for 3d printing," *Journal of Computer Science and Technology*, vol. 31, no. 3, pp. 439–449, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11390-016-1638-2 2.2

[15] M. Mukhtar, E. Akyrek, T. Kalganova, and N. Lesne, "Control of 3d printed ambidextrous robot hand actuated by pneumatic artificial muscles," in *SAI Intelligent Systems Conference (IntelliSys), 2015*, Nov 2015, pp. 290–300. 2.2

[16] M. Lapeyre, P. Rouanet, J. Grizou, S. N'Guyen, A. L. Falher, F. Depraetre, and P. Y. Oudeyer, "Poppy: Open source 3d printed robot for experiments in developmental robotics," in *4th International Conference on Development and Learning and on Epigenetic Robotics*, Oct 2014, pp. 173–174. 2.2

[17] M. Lapeyre, S. N'Guyen, A. L. Falher, and P. Y. Oudeyer, "Rapid morphological exploration with the poppy humanoid platform," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov 2014, pp. 959–966. 2.2

[18] M. Lapeyre, P. Rouanet, and P. Y. Oudeyer, "Poppy humanoid platform: Experimental evaluation of the role of a bio-inspired thigh shape," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct 2013, pp. 376–383. 2.2

[19] h. oehm. (2013, March) Openscad micro servo robot arm. [Online]. Available: http://www.thingiverse.com/thing:65081 2.2

[20] C. Franciscone. (2015, September) Eezybotarm. [Online]. Available: http://www.thingiverse.com/thing:1015238 2.2

[21] C. Arnø. (2014, May) 4 axis robotic arm. [Online]. Available: http://hacknorway.com/wordpress/4-axis-robotic-arm/ 2.2

[22] angelLM. (2016) Thor. [Online]. Available: https://hackaday.io/project/12989-thor 2.2

[23] Zortrax. (2016) Get your free 3d files for the robotic arm. [Online]. Available: https://zortrax.com/free-robotic-arm-files/ 2.2

[24] BCN3D. (2016, July) Bcn3d moveo a fully open source 3d printed robot arm. [Online]. Available: https://www.bcn3dtechnologies.com/en/bcn3d-moveo-the-future-of-learning/ 2.2

[25] S. Robotics. (2016, March) Most advanced robotics simulation software overview. [Online]. Available: https://www.smashingrobotics.com/most-advanced-and-used-robotics-simulation-software/ 2.3

[26] Vex. (2016) why vex iq. [Online]. Available: http://www.vexrobotics.com/vexiq/why-vexiq 2.3

[27] J. F. M. Lee and J. A. Buitrago, “Map generation and localization for a lego nxt robot,” in *Automatic Control (CCAC), 2015 IEEE 2nd Colombian Conference on*, Oct 2015, pp. 1–5. 2.3

[28] M. Pinto, A. P. Moreira, and A. Matos, "Localization of mobile robots using an extended kalman filter in a lego nxt," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 135–144, Feb 2012. 2.3

[29] . Hmori, J. Lengyel, and B. Resk, "3dof drawing robot using lego-nxt," in *2011 15th IEEE International Conference on Intelligent Engineering Systems*, June 2011, pp. 293–295. 2.3

[30] J. M. G. de Gabriel, A. Mandow, J. Fernandez-Lozano, and A. J. Garcia-Cerezo, "Using lego nxt mobile robots with labview for undergraduate courses on mechatronics," *IEEE Transactions on Education*, vol. 54, no. 1, pp. 41–47, Feb 2011. 2.3

[31] C. Ltd. (2016) Buy webots. [Online]. Available: http://www.cyberbotics.com/buy 2.3

[32] O. S. R. Foundation. (2014) Gazebo tutorials. [Online]. Available: http://gazebosim.org/tutorials?cat=install 2.3

[33] DHood. (2016, May) Debian install of ros kinetic. [Online]. Available: http://wiki.ros.org/kinetic/Installation/Debian 2.3

[34] D. Rivas, M. Alvarez, P. Velasco, J. Mamarandi, J. L. Carrillo-Medina, V. Bautista, O. Galarza, P. Reyes, M. Erazo, M. Prez, and M. Huerta, "Bracon: Control system for a robotic arm with 6 degrees of freedom for education systems," in *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, Feb 2015, pp. 358–363. 2.3

[35] M. Ferguson, N. Webb, and T. Strzalkowski, "Nelson: A low-cost social robot for research and education," in *Proceedings of the 42Nd*

ACM Technical Symposium on Computer Science Education, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 225–230. [Online]. Available: http://doi.acm.org.ezproxy.southern.edu/10.1145/1953163.1953230 2.3

[36] Y. Hold-Geoffroy, M. A. Gardner, C. Gagn, M. Latulippe, and P. Gigure, "ros4mat: A matlab programming interface for remote operations of ros-based robotic devices in an educational context," in *Computer and Robot Vision (CRV), 2013 International Conference on*, May 2013, pp. 242–248. 2.3

[37] D. Herrmann. (2013, December) Xwiimote. [Online]. Available: http://dvdhrm.github.io/xwiimote/ 3.2.1.6

[38] M. Purvis. (2015, October) Unittesting. [Online]. Available: http://wiki.ros.org/UnitTesting 4.2

[39] E. R. Doering, "Electronics lab bench in a laptop: using electronics workbench 174; to enhance learning in an introductory circuits course," in *Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change. Proceedings.*, vol. 1, Nov 1997, pp. 18–21 vol.1.

[40] BCN3D. (2016, October) Bcn3d-moveo: Open source 3d printed robotic arm for educational purposes. [Online]. Available: https://github.com/BCN3D/BCN3D-Moveo