Project R.I.C.K.

(University Senior Design Project)

An all-in-one system that takes **input** directly from a person's arm/finger **movements** from **camera sensors** to be translated into **machine code** as output controls for a variety of **robotic arms** in a rapid and scalable fashion

December 5th, 2017
Zachary Campanella
Mohammad Daraghmeh
Thanh Pham
Cole Pickering

Contents

1. Overview

1.1. Customer Needs Statement

Approximately four decades ago, industrial robots began transforming the world of manufacturing on a global level to supplant human workers. By 2020, the robotics industry is predicted to surpass $145.4 billion from its 2016 standing at around $71 billion worldwide.[1] During the first half of 2017, a total of 19,331 robots valued at approximately $1.031 billion were sold in North America a current record high[2]. This increase in the number of robots will create a demand for simple and intuitive programming procedures. In order for assembly lines to be robust, systems must be connected and modular. A platform is needed to allow users to seamlessly translate human action into robot code to perform singular or repetitive tasks without the need of intensive work, coding knowledge, or specialized workers.

1.2. Objective Statement

The objective of this project is to design and prototype a device that will convert human actions into robot code (G-code). These user actions will be caputed using input from a IR depth-finding camera and motion sensing technology to intuitively control an industrial style robotic arm. The device will record these movements and translate them to G-code a programming language for CNC (Computer Numerical Control) machines

[1]fortune.com/2016/02/24/robotics-market-multi-billion-boom/
[2]vision-systems.com/articles/2017/08/robotics-and-machine-vision-sales-reaching-new-heights-in-north-america-in-2017

so that the robotic arm is swiftly and accurately programmed. The usage of the device will reduce the amount of time it takes to add or repurpose a robotic arm in an assembly line.

1.3. Description

The Microsoft Kinect was primarily used to detect natural arm movements and hand gestures. An Arduino Mega microcontroller coupled with a RAMPS stepper driver shield was used to take in these movements and transmit them over USB which was then interpreted on a Raspberry Pi 3 microprocessor. The microprocessor responsibility at the point was to apply custom filtering and smoothing to the sensory input. This refined sensory data done by software was then outputted to the 6-axis robotic arm in G-code. This robotic arm was 3D printed and assembled using an open-source design kit. The kit used stepper motors in order to have precise control with high torque in order to pick up heavy items such as a water bottle.

1.4. Marketing Picture

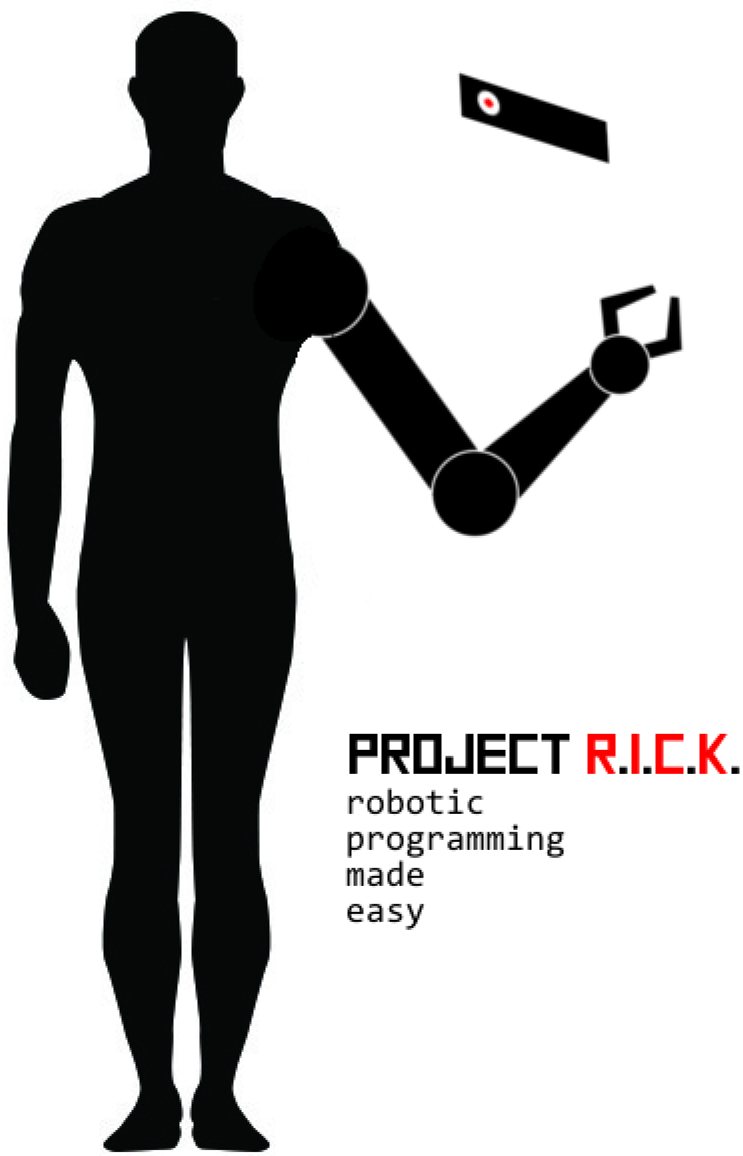

Figure 1: Marketing Picture

2. Requirements Specification

2.1. Customer Needs

1. The system must be able to provide real-time conversion of the user's arm movements/commands to an acceptable robot language.

2. The system must be controlled with natural and simple motions and gestures from the user such as extending the arm or grabbing an object.

3. The system must be moderately portable.

4. The system must be able to lift objects of weight comparable to those needed in industry, such as computer components, soda bottles, and etc.

5. The system must be intuitive to learn.

6. The system must be easy to assemble and disassemble.

7. The system must be able to accurately detect the same gesture with a high accuracy rate.

8. The system must be able to accommodate an array of environments.

9. The system must be able to record and repeat tasks.

10. The system must be safe to use.

2.2. Engineering Specifications

Table 1: Engineering Requirements

Marketing Requirements	Engineering Requirements	Justification
1,2,7	**A**. The system must be able to map hand and arm gestures to specific commands.	The user gestures required should be an extension of the user's natural arm and hand manipulations. This minimizes the amount of learning the user requires to manipulate the arm.
3,4,8	**B**. The system must process information (hand gestures) in an array of environmental conditions.	The environment is key in order for the sensor system to register the user's gestures/motion.
1	**C**. The system must follow the user's movements within 1 sec.	In order for the system to give accurate visual feedback, it must operate in real time.
1,9	**D**. The system repeatedly reproduce a user input path.	The purpose of the system is to allow a robotic arm to be easily programmed, so it is important that the arm can repeat its motion.
1, 10	**E**. The system must satisfy OSHA directive STD 01-12-002. (https://goo.gl/ULlqMj)	Robotic arms are a large safety concern in any factory setting, so it is important that the robotic arm does not move in an unexpected way to not cause damage or injure people.
1, 2	**F**. The system's motors must have consistent and accurate stepping movement and hold positions with a payload.	The movement should be very precise without jitteriness for real-time movement replication.
3	**G**. The system must not require multiple tools(e.g. Multiple screw bits) to assemble, replace, or disassemble.	Reducing the startup time and weight would allow for higher mobility.
4	**H**. The system must have the robotic arm connected to the microcontroller.	Reducing the number of failure points and complexity will also reduce the amount of knowledge to set up.

5	**I**. The system must be able to be learned within 60 min.	The system is meant to be intuitive so it should be easy to learn.

3. Concept Selection

3.1. Survey of Existing Systems

Several existing systems were surveyed to analyze their advantages and limitations. All use a different set of sensors to transfer the user's arm movements into directions for a robotic arm. The sensor systems examined are Siemen's ROBCAD and teaching pendants.

1. Siemen's ROBCAD provides software to simulate and program a robotic arm from a Windows computer station. The programing of the robot is done in a virtual 3D environment, without the robotic arm present. This is specialized software that is expensive and time consuming to work with. Special training in the software is required. (https://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/robotics/robcad.shtml)

   Relation: This is a direct competitor to the Project R.I.C.K programming method. It is difficult to work with, and is not agile, requiring the simulation environment to be changed whenever changes are made on the production line. The system is possibly mobile, if using a laptop computer.

2. Teaching Pendants, like the ones built by Denso Robotics, provide plug-in control to program robots in a motion by motion process. These are dust-proof, hardened devices with buttons and an LCD screen. At under 2.2 pounds they are very portable, meant to be plugged directly into the robot. (http://densorobotics.com/products/teaching-pendants/spec)

   Relation: Another direct competitor to Project R.I.C.K. These pendants take some time to program each robotic motion. Each axis of the robot must be individually actuated to get to the correct position for the step. These units are not wireless, relying on a 4 to 12m cable for both power and data capability. Pendants have to translation of button presses that directly control arm movement.

3.2. Sensor

Based on the survey of existing systems, two major concept requirements were decided. First, the sensor system should be wireless. In general, wireless sensors are needed in real world scenarios for the robotic arm being controlled may be beyond the user's physical reach. Also, a wireless system for the most part can be very user friendly if done right. The second requirement was to have multiple inputs as needed in order to remove human noise, environmental factors, and provide a more reliable stream of data for processing.

The following Pugh table (Table 2) below was generated in order to allow for a more specific sensor system comparison. Weights were assigned to each criteria based on the requirements of this project. The Microsoft Kinect scored the highest on the Pugh tables with a total score of four. When observing the table in more detail, for the most part the pros for the Kinect were the cons for the flex glove and vice versa.

Table 2: Pugh Analysis for Sensor Options

			Microsoft Kinect		**Flex Glove**	
Criteria	**Weight (1-10)**	**Base**		**Value**		**Value**
Cost	7	Under $100	$80	1	~$125	-1
Sensors Used	7	Multiple Sources	RGB camera, depth sensor, microphone	1	EMG sensors, 3-axis gyroscope, accelerometer, flex sensors	1
Recognition	10	Arm Motions & Hand Motions	full-body 3D motion capture, voice recognition	1	Bending, rotating,grip, 5 figure gestures	0
Communication	5	Wireless	USB 2.0	-1	Bluetooth	1
Data Interface	5	Windows	Windows, Linux (Opensource)	1	Windows, Linux	1
User Interface	10	Wireless/Portability	Stationary and pointed at subject	-1	Attaches to arm	1
Battery Life	3	8 Hours (Business Day)	Wired	1	1 Day	1
Complexity	8	Easy to integrate to the project has a whole	Hardware is all in one package. Software adds a layer of complexity	0	Adds the complexity of individual parts and sensor data	-1
Additional Addons/Parts	4	Does not require addons or external parts to meet its needed purpose	All-inclusive	1	Battery bank, glove fabric, and etc	-1
		Total		**4**		**2**

3.2.1. Selection of Sensor System

The Kinect as a sensor system satisfies the requirements needed to achieve the project's main goal of providing a seamless control of a variety of robotic arms. The Kinect is well documented, readily available, and similar projects to this have been published online for our reference. In addition, the Kinect is fairly portable and due to the lack of customer demand the Kinect retail value has decreased greatly. The Kinect also provides a natural and simple method for customers to manipulate a robotic arm for traditionally methods require coding and technical backgrounds. Lastly, the most important need of this sensor system was to be able gather data about the user's movements and hand gestures accurately and responsively. The Microsoft Kinect not only satisfies this important need but all these mentioned previously.

3.3. Computing System

The specifications gathered from the analysis of other systems show that a microcomputer will be the best option for the computing core of the Project R.I.C.K. system. There are many different types of microcomputers, and the variety of features and prices allows you to make a choice that is very close to what your application needs. The single-board computers selected for comparison are some of the newest and most popular in the maker community, allowing a vast array of expertise available on the internet.

3.3.1. Selection of Computing System

Four of the most common single board computer were compared based on their specifications. Processing power was weighted highest, because of the uncertainty of how much power it will take to process large amounts of data and have close to real time translation. Wireless communications systems and USB capability were also requirements of the system. Table 3 and 4 show the comparison done between the systems.

Table 3: Pugh Analysis for Computing Core Options 1 and 2

		BeagleBone Black		Raspberry Pi 3	
Criteria	**Weight**		**Value**		**Value**
Cost	5	54.95	0	$35	1
Processing Power	8	1 GHz ARM Cortex-A8	-1	1.2 GHz ARM Cortex-A53	1
Number of USB Ports	6	1x USBA	-1	4x USBA 2.0	1
Wireless capability	5	Ethernet only	-1	Bluetooth 4.1, BLE, 2.4GHz 80.11n wireless	1
Amount of ram	7	512MB DDR3	-1	1GB LPPDDR2	0
Size / weight	1	86.4 x 53.3 40g	1	85.6 × 56.5 × 17.0 45g	0
Total			-9		24

Table 4: Pugh Analysis for Computing Core Options 3 and 4

		UDOO NEO Extended		ODROID-C2	
Criteria	**Weight**		**Value**		**Value**
Cost	5	59.90	-1	$46	0
Processing Power	8	ARM Cortex-A8 and Cortex-M4	0	1.5 GHz ARM Cortex-A53	1
Number of USB Ports	6	1x USBA, 1x USB OTG	0	4x USBA 2.0	1
Wireless capability	5	BLE, Wi-Fi 802.11 b/g/n	1	Ethernet Only	-1
Amount of ram	7	1GB	0	2GB DDR3	1
Size / weight	1	89mm x59mm	0	85 × 56 × 17.0	0
Total			0		16

The Raspberry Pi 3 was selected as the computing core of Project R.I.C.K because it proved to have the most IO ports, utility, and processing power for the lowest cost.

3.4. Movement Control Algorithm

There were three point-to-point control methods and two continuous path control methods that were investigated. The control methods needed to be able to calculate the speed, travel time, and position of each axis in the robotic arm using the current position of the robotic arm and the user input position. It is imperative that the calculations can be completed very quickly so that the arm can closely follow the user input movements.

The movements that are generated, in addition to being quickly calculated, need to give the user good visual feedback of how and where the robotic arm is moving so the user is able to correct any errors with the path. In addition to visual feedback, a predictable path is important so that any person or object near the arm is not in physical danger.

3.4.1. Selection of Movement Control Algorithm

The point-to-point methods that were considered were one joint at a time, slew motion, and joint interpolation. The continuous path methods that were considered were linear interpolation and circular interpolation. The continuous path methods were quickly discounted as they function the best knowing the whole path of the robotic arm and require more complex calculations. The point-to-point methods fit the design of the system more as they require only a starting and end position to generate movement information.

The one joint at a time method moves the joints in a sequential manner, which means the visual feedback is slow and awkward. This method takes the longest time to finish a movement, but requires the least amount of simultaneous power.

The slew motion begins the movement for each axis at the same time, but the movement of each axis ends at different times. This allows the arm to move from one position to another faster than the one joint at a time method. The slew motion also more closely follows the overall direction that the arm is moving. The drawback of this method is that it is more complicated to implement than the previously mentioned method and it has a higher simultaneous power consumption.

The joint interpolation method begins the movement for each axis at the same time and each joint finishes movement at the same time. This method was selected to be implemented. The reason for this is that is has the same travel time as the slew motion, but gives better visual feedback as each axis starts and stops movement at the same time. This method is more complicated to implement than the slew motion method and has a higher simultaneous power consumption than the one joint at a time method, but should have smoother power curve as the axes that require little movement will have longer ramp up and ramp down times than in slew motion.

3.5. Robotic Arm

Following the survey of existing systems, four major concept requirements were established. First, the robotic arm needs to have a minimum of 5, and preferably 6, degrees of freedom. This would allow the robotic arm to mimic human arm movement precisely without compromising the robot's region of support. Second, the robotic arm needs to run on stepper motors because that will allow the arm to have accurate and precise mimicking of movement. Servos could be used in place of the stepper motors for a larger load capacity, but would lose the fine motor control. Third, the robotic arm design needs to be easily modifiable which means that replacing an arm component would be easily replicated with little wait time or cost. Fourth, the robotic arm must be compact which could include integrating a portion of the robotic arm's base as the location of the mounted microcontroller to increase weight and minimalistic design.

3.5.1. Selection of Robotic Arm

The following Pugh table (Table 5) below was generated in order to allow for a quantitative detailed robotic arm comparison. Weights were assigned to each criterion based on the requirements of this project.

Table 5: Pugh Analysis of Robotic Arm

			SainsSmart		**MOVEO**	
Criteria	Weight (0-1)	Base		Value		Value
Cost	.2	Under $200	$140	5	50(Motors) + 30(Plastic)	4
Degree of Freedom	.3	5	6	5	5	4
Load Capacity	.1	172g	500g	5	?	4
Torque	.2	20kg	30kg	2	4400g/cm	4
Motor Type	.3	Stepper	Servo	4	Stepper	5
Total				**4.1**		**4.3**

The table compared various key features required for the robotic arm. The MCN3D MOVEO scored the highest on the Pugh tables with a total score of 4.3. While the load capacity and torque are still to be determined for this design, the 3D printed design would be the the best option for its low cost and higher customizability for its motors.

4. Design

4.1. Overall System

The system is divided into subsystems that work together to provide the functionality described in the engineering specifications. The four systems are the sensor system, control system, the microcontroller, and the robotic arm. The overall design consists of the Kinect being used as the all-inclusive sensory input device. The captured user’s arm and hand gesture data will be sent directly over CAT5 or a wireless medium such as Wi-Fi Direct to the Raspberry Pi 3 microprocessor. Once the data received is processed through custom filtering and smoothing done via custom written software the refined sensory data will then be outputted in G-code. The final step would be to push this G-code directly to the robotic arm in a 1:1 in real time speed.

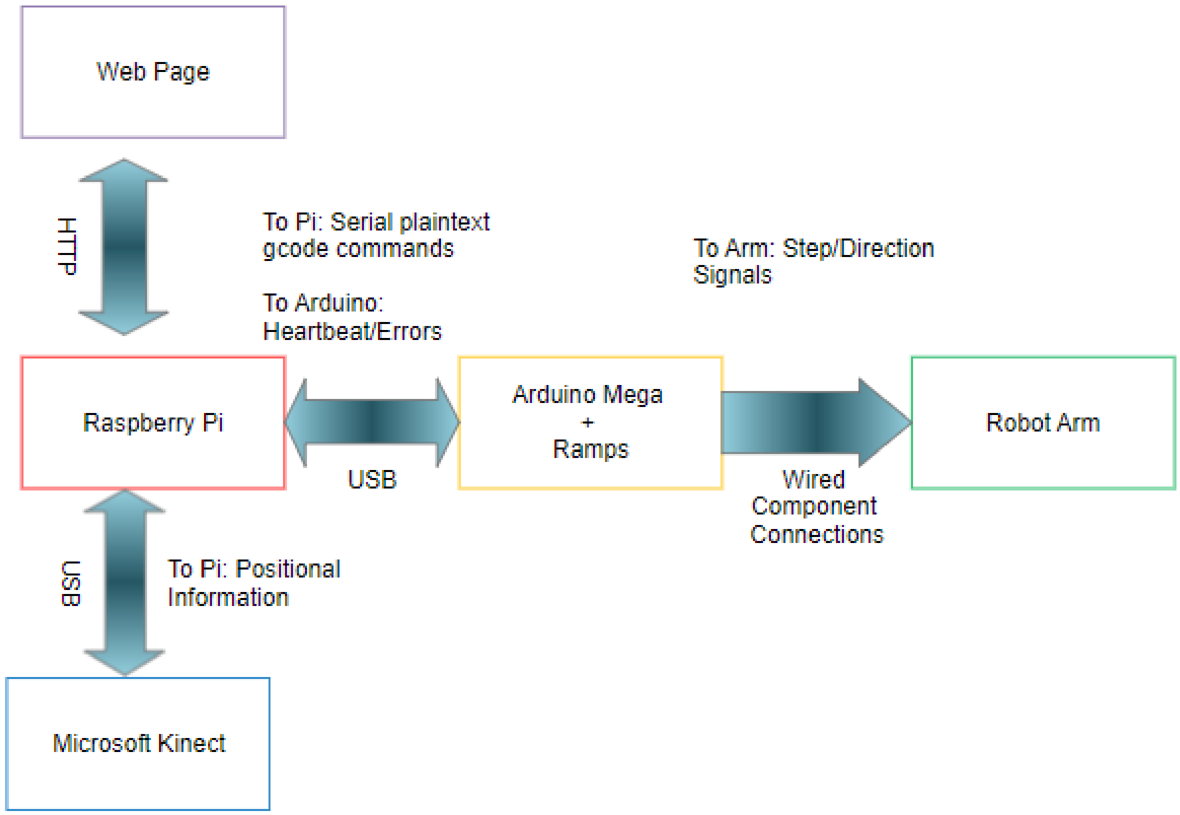

Figure 2. System Diagram

4.2. Robotic Arm Design

The Robotic arm design is an open source design from BCN3D Technologies. The design uses a 3D printed structure and stepper motors to provide high torque, high accuracy movement. Using technologies borrowed from the 3D printing community, the stepper motors are controlled with a RAMPS 1.4 board, a shield that mounts on the Arduino Mega. The Arduino Mega provides a large number of I/O that makes it possible to control all five required stepper motors. Figure 3 shows the approximate wiring required for the stepper motors on the Ramps board.

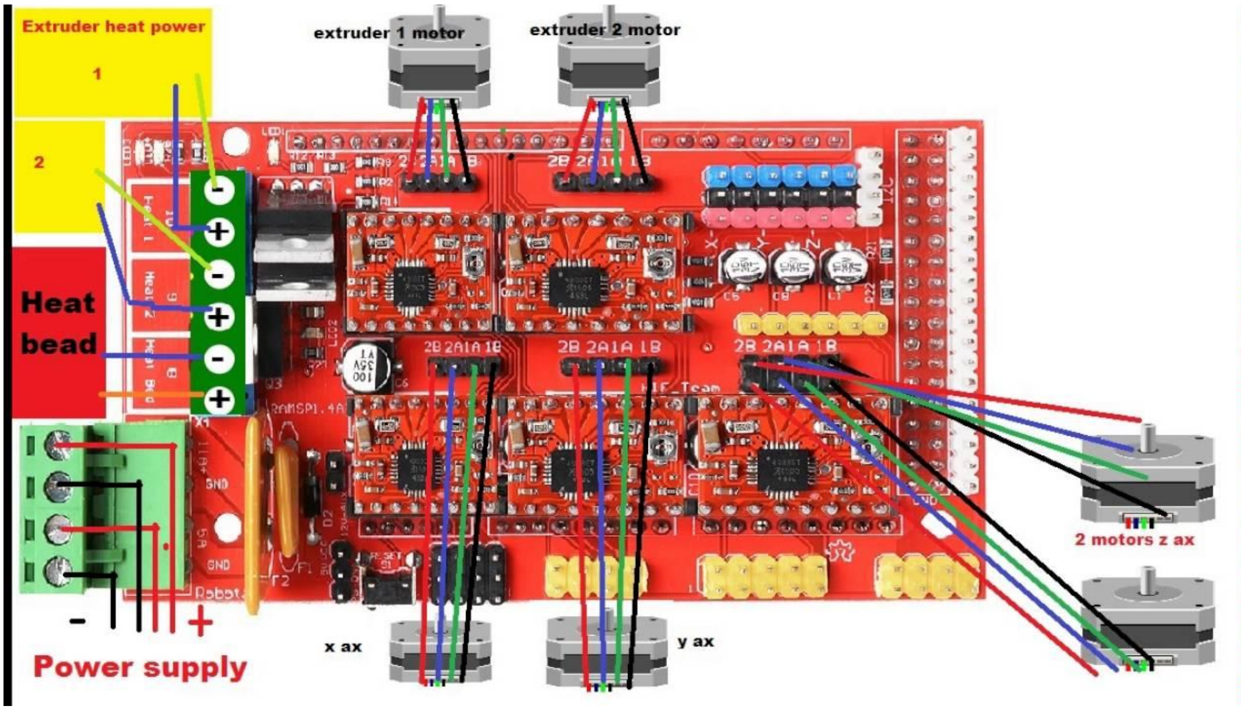

Figure 3. Ramps 1.4 wiring diagram (https://www.youtube.com/watch?v=U-VyUV3k6xI**)**

The Arduino Mega will be running a modified version of the open source Repetier firmware. This firmware is designed to translate G-code into control signals via serial input. The Repetier firmware provides a commands loop base that movement code can be easily plugged into. The motion will be controlled using Bresenham's line algorithm to create smooth multi access straight line movements. This subsystem does not have any direct user control. Inputs will be taken in via serial G-code from the Raspberry Pi.

4.3. Raspberry Pi

The software running on the Raspberry Pi takes in input from the Microsoft Kinect via USB. The OpenKinect open source library will be used to read the data brought in from the Kinect. The software will translate Kinect data to machine code, taking into account axis limits to allow the arm to move to all points in it’s movement envelope. The Raspberry Pi will kinect to the robot controller via serial, sending commands as soon as they are available to send. To be as user friendly as possible, the Raspberry Pi software should run on power-up, with no need for user interaction. Each peripheral unit should be detected and connect when they are plugged in.

4.4. Kinect

The Kinect takes user inputs through its sensors and sends them to the Raspberry Pi. The Kinect will need to be mounted such that it provides the greatest detail in user movement. Lighting may be required to make the Kinect function in any location.

4.5. User Interface/Control

The only user interface required is gesture control through the Microsoft Kinect. The software needs to be designed in such a way that the system starts working when power is applied and each subsystem is plugged into the Raspberry Pi. There will need to be an ancillary button that serves as an emergency stop for the robot arm that provides a kill-switch in case of danger to the operator or others.

4.6. Engineering Standards

Table 6 shows the engineering standards used in this project.

Table 6: Engineering Standards used

Standard	Organization	Usage
USB	USB Implementers Forum	The connections between the Raspberry Pi, Microsoft Kinect and Arduino Mega will be made via USB
G-code (*RS-274)*	Many, modified per implementation	Numerical control language to be used to communicate movement from Raspberry Pi to Robotic Arm

4.7. Multidisciplinary Aspects

The robotic arm is a mix of computer engineering with low level programming and mechatronics, and mechanical engineering in the structure of the arm. The algorithms translating user motion to robotic motion is a software engineering or computer science disciplinary area.

4.8. Background

The relevant class work completed that has been and will be used in this project comes from Interface and Digital Electronics, CMPE-460. A complete system, from input sensors to signal conditioning to output were stressed in the Interface and Digital Electronics lectures. The performance of algorithms written in low level software languages was a main tenet of CMPE-380, Applied Programming. The control theory taught in that class will be applied to the control of the robotic arm. Outside 3D printer experience had by team members will help with not only printing the robotic arm shell, but also stepper control and pathfinding.

4.9. Outside Contributors

There are no outside contributors in this project.

5. Constraints and Considerations

5.1. Extensibility

This project is a proof of concept and will not have perfect algorithms and response speed. The algorithms used will be able to be refined for more efficient movement, as well as speed of completion. There is also room available to make the handling of output G-code more flexible, allowing for various forms of numeric control used by companies.

5.2. Manufacturability

The only manufacturing that would be required is wiring together the components and loading the system with the required software. All components are able to be plugged into each other via USB, which allows for user assembly. The robotic arm is not a part of the system that would be sold to a consumer as it only exists to showcase the other components.

5.3. Reliability

The Raspberry Pi and Microsoft kinect are consumer products that are extensively tested before being sold. This makes them very reliable, but there is possible need for active cooling on the Raspberry Pi to keep it cool during processor intensive calculations. The biggest concern with regards to reliability is with the robot arm. The arm is a modular system which will allow individual parts to be replaced quickly and easily if failure occurs.

5.4. Others

The times spent between each portion of the project is a large consideration that needs to be made. The robotic arm system is only an ability to show off the motion control and translation of the Kinect and software algorithms. Construction, testing and tweaking of the arm needs to be completed very early in the process so focus can be placed on the main portions of the project.

6. Bill of Materials

The robotic arm is an off the shelf open-source design from Thingiverse. The structure of the robot will be 3D printed. The Raspberry Pi, and PSU are free to the team from items that they already own. The stepper motors come from omc-stepperonline to provide reliable, quick shipping. These steppers needed to be sourced individually by matching the stepper motor specifications from the Moveo build of materials.

Table 8: Robotic Arm Cost Breakdown

Part Description	**Cost ($)**	**Team Cost ($)**	**Availability**
Arduino Mega 2560	10.00	15.00	Banggood.com
Stepper/Servo - Various Types(7)	95.00	95.00	omc-stepperonline.com
RAMPS v1.4	10.00	10.00	Banggood.com
PSU	150.00	0.00	Personal Item
Screws/Bolts	30.00	30.00	Banggood.com | Home Depot
Raspberry Pi 3	39.95	0.00	Personal Item
Microsoft Kinect V2	99.99	49.99	www.amazon.com
Total	483.93	247.93	

The total project cost totals at $287.93. This leaves a healthy margin of error within the $400 budget. The cost calculated is for the major components, but additional smaller items may be needed. There is $112 left for these expenses.

7. Testing Strategy

7.1. Unit Tests

The following unit tests for each subsystem: the Microsoft Kinect, the control system, the microcontroller, and the robotic arm. The unit tests are used to verify individual functional performance. There are different number of unit tests for each subsystem because certain functionality has been guaranteed from commercial products.

Test Writer: Mohammad Daraghmeh						
Test Case Name:	Out of the Box Kinect Test				**Test ID#:**	1001
Description:	Verify that the Microsoft Kinect functions as advertised using the Kinect's open SDK. Ultimately allowing to check that gestures are recognized.				**Type:**	☐ white Box ☑ black box
Setup:	Have Kinect powered on and connected to a computer with preinstalled Kinect SDK.					
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Updating the Kinect's software.	SDK should prompt for update that will end with successful update prompt.	✓			

2	Positioning the Kinect sensor to accurately detect the motion of the user's body.	Green check mark from SDK proving that placement and distance from subject (user) is correct.	✓			
3	Calibrating the Kinect sensor via Kinect Tuner within SDK.	Kinect calibration card registered correctly generating an alert that the calibration is complete.	✓			
Overall Test Result			✓			

Test Writer: Mohammad Daraghmeh			
Test Case Name:	Kinect Arm Detection	**Test ID#:**	1002
Description:	Kinect camera can detect and model human arm and hand.	**Type:**	☑ white Box ☐ black box
Setup:	Have Kinect powered on and connected to a computer with preinstalled Kinect SDK.		

Step	**Action**	**Expected Result**	Pass	Fail	N/A	**Comments**
1	No arm movement.	Data output should produce coordinates of arm and hand in 3D space.	✓			
2	Move single arm straight up and then down.	Modeled arm from the recognized kinect "skeleton" will mimic the action described as well as provide raw data such as the xyz coordinates of the tracked arm that was moving upwards and	✓			

		then downwards, respectively.				
3	Move single arm to left and then right.	Modeled arm from the recognized kinect "skeleton" will mimic the action described as well as provide raw data such as the xyz coordinates of the tracked arm that was moving left and then right, respectively.	✓			
4	Simulate a human grip on an object.	Modeled hand is gripping an object.	✓			
5	No visible arm or hand.	Modeled hand is not created.	✓			
6	Move arm in the X, Y, and Z directions at various speeds.	Data output should match direction of user's arm movement and correct coordinates of arm and hand in 3D space.	✓			
Overall Test Result			✓			

Test Writer: Thanh Pham

Test Case Name:	Out of the Box Motor PWM Test	**Test ID#:**	1003
Description:	Ensure all stepper motors can rotate properly.	**Type:**	☑ white Box ☐ black box
Setup:	Motor wires connected to RAMPS shield and then connected to Arduino.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Send	Motor moves at maximum speed.	✓			

	maximum PWM signal to motor.					
2	Send over maximum PWM signal to motor.	Motor moves at maximum speed.	✓			
3	Send a pulse of pwm signals to motor.	Motor moves to speed of pulse.	✓			
Overall Test Result			✓			

Test Writer: Zachary Campanella			
Test Case Name:	Movement Control Algorithm Validity	**Test ID#:**	1004
Description:	Feeds data to the algorithm to see the calculated interim steps. Checks if each movement makes sense and each motor location is possible based on input data (some motors have to be the same distance from each other as when they started).	**Type:**	☑ white Box ☐ black box
Setup:	Algorithm is compiled in digital test environment in test mode.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Give current location.	Outputs current location of robotic arm.	✓			
2	Give a small	Outputs the locations of the arm in very small steps. Locations of	✓			

	change in tool position up then down.	motors should be valid based upon original location data and final locations should be valid for the input change.				
3	Give a small change in tool position left then right.	Outputs the locations of the arm in very small steps. Locations of motors should be valid based upon original location data and final locations should be valid for the input change.	✓			
4	Give a small change in tool position tilt up then down.	Outputs the locations of the arm in very small steps. Locations of motors should be valid based upon original location data and final locations should be valid for the input change.	✓			
5	Give a small change in tool position rotate clockwise then counterclo ckwise.	Outputs the locations of the arm in very small steps. Locations of motors should be valid based upon original location data and final locations should be valid for the input change.	✓			
Overall Test Result			✓			

Test Case Name:	Movement Control Algorithm Bounds	**Test ID#:**	1005
Description:	Gives algorithm locations outside of the designated operating area to make sure the arm does not exit the operating area.	**Type:**	☑ white Box ☐ black box
Setup:	Algorithm is compiled in digital test environment in test mode.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Give current location.	Outputs current location of robotic arm.	✓			
2	Give operating area.	Uses operating area as hard bounds for calculations.	✓			
3	Give a tool location that is outside of bounds.	Algorithm should move as close to the location as possible without leaving bounds. Should inform that it was unable to fully complete the task.	✓			
4	Give a tool orientation on the bounds facing out of bounds.	Should output locations of motors that are inbound and accurate for the given movement information.	✓			
5	Give a tool orientation in the same location as Step 4 but with an opposite orientation.	Algorithm should move as close to the location as possible without having any motor leave the operating area. Should inform that it was unable to fully complete the task.	✓			
Overall Test Result						

Test Writer: Zachary Campanella			
Test Case Name:	Motor Waveforms	**Test ID#:**	1006
Description:	Feeds data to the RAMPS board and measures the output waveforms.	**Type:**	☑ white Box ☐ black box

Setup:	Ensure proper connection of oscilloscope to RAMPS board. Plug RAMPS board into computer via USB.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Send ‘G1’ command stepper 1 move 5 mm forwards	Output waveform on the oscilloscope matches the waveform of a stepper moving forward.	✓			
2	Send ‘G1’ command stepper 1 move 5 mm backwards	Output waveform on the oscilloscope matches the waveform of a stepper moving backwards.	✓			
Overall Test Result			✓			

Test Writer: Cole Pickering			
Test Case Name:	Single Robotic Arm Accuracy	**Test ID#:**	1007
Description:	Send commands to robot arm controller via Serial Communication to test motors.	**Type:**	☑ white Box ☐ black box
Setup:	Ensure proper connection of stepper motors to RAMPS board. Plug RAMPS board into computer via USB. Connect computer to Arduino via serial.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Send ‘Home’ command.	Robot moves to ‘Home’ position	✓			
2	Send ‘G1’ command stepper 1	Stepper 1 moves forwards 2mm	✓			

	move 2 mm forwards					
3	Send ‘G1’ command stepper 1 move 5 mm forwards	Stepper 1 moves forwards 5mm	✓			
4	Send ‘G1’ command stepper 1 move 5 mm backwards	Stepper 1 moves backwards 5mm	✓			
Overall Test Result			✓			

Test Writer: Cole Pickering						
Test Case Name:	Multiple Robotic Arm movement				**Test ID#:**	1008
Description:	Send commands to robot arm controller via Serial Communication to test motors.				**Type:**	☑ white Box ☐ black box
Setup:	Ensure proper connection of stepper motors to RAMPS board. Plug RAMPS board into computer via USB. Connect computer to Arduino via serial.					
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Send ‘Home’ command.	Robot moves to ‘Home’ position	✓			
2	Send ‘G1’ command stepper 1 to mid position	Stepper 1 moves to midpoint position.	✓			
3	Send ‘G1’ command	Stepper 2 moves to midpoint position.	✓			

	stepper 2 to mid position					
4	Send ‘G1’ command stepper 3 to mid position	Stepper 3 moves to midpoint position.	✓			
5	Send ‘G1’ command stepper 4 to mid position	Stepper 4 moves to midpoint position.	✓			
6	Send ‘G1’ command stepper 1 to mid position	Stepper 5 moves to midpoint position.	✓			
7	Send ‘G1’ command stepper 1 to mid position	Stepper 6 moves to midpoint position.	✓			
Overall Test Result			✓			

7.2. Integration Tests

The following integration tests are designed to ensure that all four subsystems work together to achieve the user’s desired input. Therefore, most of the integration tests require that all the unit tests have passed for each of the subsystems.

Test Writer: Thanh Pham			
Test Case Name:	Save Task Script	**Test ID#:**	2001
Description:	Arduino is able to save a task script of movement commands from code and human arm.	**Type:**	☑ white Box ☐ black box

Setup:		Kinect is connected to Raspberry Pi.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Save valid movement commands as code.	Robot arm moves as per specification of command.	✓			
2	Save invalid movement commands as code.	Robot arm does not move and the script logs error.	✓			
3	Human movement is saved as code	Robot commands were correctly transcribed from human movement.	✓			
Overall Test Result			✓			

Test Writer: Zachary Campanella						
Test Case Name:		Safe Robotic Movement			**Test ID#:**	2002
Description:		Issues move commands to the robotic arm that are outside of its operating area.			**Type:**	☑ white Box ☐ black box
Setup:		Robotic arm is securely connected to RAMPs board which is connected to the arduino which is connected to a computer over serial.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Give operating area.	Uses operating area as hard bounds for calculations.	✓			

2	Give a tool location that is outside of bounds.	Arm should move as close to the location as possible without leaving bounds. Should inform that it was unable to fully complete the task.	✓			
3	Give a tool orientation on the bounds facing out of bounds.	Arm should move to the edge of the operating area without anything leaving the bounds.	✓			
4	Give a tool orientation in the same location as Step 4 but with an opposite orientation.	Arm should move as close as possible to intended location without leaving bounds.Should inform that it was unable to fully complete the task.	✓			
Overall Test Result			✓			

7.3. Acceptance Tests

The following acceptance tests verify that the final system meets the engineering requirements.

Test Writer: Mohammad Daraghmeh & Cole Pickering			
Test Case Name:	Mirrored Movement	**Test ID#:**	3001
Description:	The movement of the arm should be accurate when compared to the movements captured by the Kinect. This test will determine if the robot arm movements in 3D space are correct with respect to user movements. As well as testing to determine if the robotic arm is able to mirror the movements of user.	**Type:**	☑ white Box ☐ black box

Setup:	Connected Microsoft Kinect, RAMPS board into Raspberry Pi via USB. Powered on Raspberry Pi.					
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Standing in front of the kinect, hold arm straight out	Arm moves to arm location.	✓			
2	Move arm to right	Arm mirrors action, moves to right	✓			
3	The user moves their arm with the Myo band.	The robotic arm moves in the same manner as the user moves their arm.	✓			
4	The user creates sporadic movement s of their arm while moving in a specific direction.	The robotic arm should smoothly move to the final position without any jerkiness.	✓			
Overall Test Result			✓			

Test Writer: Thanh Pham			
Test Case Name:	Repeat Task	**Test ID#:**	3002
Description:	Robot arm can repeat a recorded task, indefinitely.	**Type:**	☑ white Box ☐ black box
	Upload task script to Raspberry Pi.		

Setup:						
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Upload blank script.	Robot arm should not move.	✓			
2	Upload simple movement script.	Robot arm repeats a simple movement, indefinitely.	✓			
3	Upload complex movement script.	Robot arm repeats a complex movement, indefinitely.	✓			
Overall Test Result			✓			

Test Writer: Mohammad Daraghmeh			
Test Case Name:	Precision	**Test ID#:**	3003
Description:	The system must be able to interact with objects within 5mm of precision.	**Type:**	☑ white Box ☐ black box
Setup:	Place a series of 5 points that several mm apart on a solid background piece of paper grid.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Using kinect controlled movement, navigate the arm to	Arm moves as directed marking all five points with a red marker.	✓			

	all five points on the grid.					
2	Measure distance between marks and points.	Marks are within 5 mm of each point.	✓			
Overall Test Result			✓			

Test Writer: Thanh Pham			
Test Case Name:	Strength & Size	**Test ID#:**	3004
Description:	The robot arm must be able to pick up different weighted and shaped objects.	**Type:**	☑ white Box ☐ black box
Setup:	Have 5 different weighted objects from pencil to water bottle.Place each object in front of robot arm.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Pick up pencil.	Robot arm is able to lift and hold pencil.	✓			
2	Pick up wallet.	Robot arm is able to lift and hold wallet.	✓			
3	Pick up phone.	Robot arm is able to lift and hold phone.		✓		
4	Pick up water bottle.	Robot arm is able to lift and hold water bottle.			✓	Due to weight and plastic material the claw was not able to grip the bottle correctly.
			✓			3 out of 4 actions

Overall Test Result				passed.

8. Risks

8.1. Risk Summary

Table 9: Risk Summary

Subsystem	Risk	Mitigated
Robot Arm	Complexity, Speed, Accuracy	Selected a design that is documented and proven to be functional. Online support available for construction.
Computing Core	Latency from input to output	Selected a microcontroller with a great power to cost ratio. Vast array of internet resources to get the greatest performance out of the Raspberry Pi 3.
Sensor System	Team members have never used the system	Large community of people working on/with an open SDK for the Microsoft Kinect.
Sensor System	High degree of complexity capturing user movement and correctly translating the movement to G-code	Largest risk faced in this project, which was mitigated by the knowledge and perseverance displayed by the team.

8.2. Robotic Arm

The Robotic Arm design is completely open-source which offers every robotic arm component in a 3D model file which can be modified to change its size or length. The ability to modify each component allows us to be more robust when needing to change our design without having to worry about buying another part. 3D printing our components would allow us to rapidly plug and play different versions of the robotic arm and would be very low-cost to do. The design can use servos and stepper motors which allows us to experimentally see how well the robotic arm would fare between the two motor types.

8.3. Computing Core

The design chosen uses an off-the-shelf microcomputer that is running Linux. The ability for the computing core to be running a true operating system allows for quickly and easily running code of any language that the programmer is comfortable with. The operating system provides us with a safety net if any problems occur in the code, as processes can recover quickly from fatal errors. By choosing a system that the team is familiar with, we can spend less time trying to figure out how to use the hardware, and more time using the hardware to do what we need to do. The Raspberry Pi is very widely used, and the internet has guides on how to complete any configuration that we might need to do on the system.

8.4. Sensor System

The Microsoft Kinect eliminates many design and technical risks because it provides an all-inclusive sensory input device. The Kinect's multiple sensors work in tandem to not only recognize and record gestures but also to reduce the probability of misclassification. In addition, because the Kinect uses an infrared depth sensor camera lighting conditions, user's skin color/clothing, and even background have little impact on the performance of this sensory system. The accuracy and the robustness make this system a versatile component that can be integrated in a variety of project designs and environments.

An additional perk to using the Kinect is the open source libraries that provide numerous avenues to equations, filters, data points, etc. This removes the risk of software development for that aspect of the system. If need be the pure raw sensor data are still available. Also, the Kinect is a fully developed retail product with forums, code repositories, and community support to assist with development and testing. Using the Kinect could save significant development time as well as provide the project with the benefit of a rigorously tested retail product.

9. Schedule

Each person will be in charge of a subsystem, but will be available for help when called upon by any other person. Deadlines for specific project areas will be set in weekly team meetings with input from entire team, but final deadline decision will come from the relevant area head. Table 10 shows the subsystem that each person will be in charge of. In addition, Table 11 contains the major and minor milestones for this project and the time period at which they were completed. Lastly, Figure 4 shows that Gant chart with the deadlines for each component of the project.

Table 10: Robotic Arm Management Plan

Management Area	Manager
Robotic Arm Construction	Thanh Pham
Translation Algorithms	Zachary Campanella
Kinect	Mohammad Daraghmeh
Robotic Arm Firmware	Cole Pickering

Table 11: Milestone Chart

Milestone	Scheduled Date	Completion Date	Name	Comments
Preliminary/Construction				
Purchase Part List	09/01/17	09/07/2017	TP	Finished
Motors	09/01/17	N/A	TP	Finished
Rods/Screws/Accessories	09/01/17	N/A	TP	Finished
Kinect	09/01/17	N/A	MD	Finished
Arduino & Pi	09/01/17	N/A	CP	Finished
Test Quality/Functionality of Microsoft Kinect on Desktop	09/08/17	N/A	MD	Finished
Print Robotic Arm	09/05/17	N/A	CP/TP	Finished
Joints	09/05/17	N/A	CP	Finished
Base	09/05/17	N/A	TP	Finished
Verify Motor Functionality	09/15/17	N/A	CP/TP	Finished

Verify motor connection to RAMPS	09/15/17	N/A	TP	Finished
Verify RAMPS to Arduino	09/19/17	N/A	CP	Finished
Construct Robotic Arm	09/22/17	N/A	CP/TP	Finished
Combine motors with 3D parts	09/22/17	N/A	TP	Finished
Attach controller and boards to base	09/25/17	N/A	CP	Finished
Unit Tests	09/28/17	N/A		
Microsoft Kinect	09/28/17	N/A	MD/ZC	Finished
Verify arm/body detection	09/28/17	N/A	MD	Finished
Verify joints mapped to motors	10/03/17	N/A	ZC	Finished
Controllers	10/01/17	N/A	CP/TP	Finished
Arduino	10/01/17	N/A	CP/TP	Finished
Verify motors turn from PWM signal	10/01/17	N/A	TP	Finished
Raspberry Pi	10/06/17	N/A	MD/ZC	Finished
Verify kinect data is saved	10/06/17	N/A	MD	Finished
Robotic Arm	10/03/17	N/A	CP/TP	Finished
Verify arm can move	10/03/17	N/A	TP	Finished
Verify grippers work	10/05/17	N/A	CP	Finished
Control Theory	09/28/17	N/A	ZC	Finished
Implementation	10/15/17	N/A	ALL	Finished
Define User Gestures for Robotic Arm	10/15/17	N/A	ALL	Finished
Complete Acceptance Tests	11/17/17	N/A	ALL	Finished

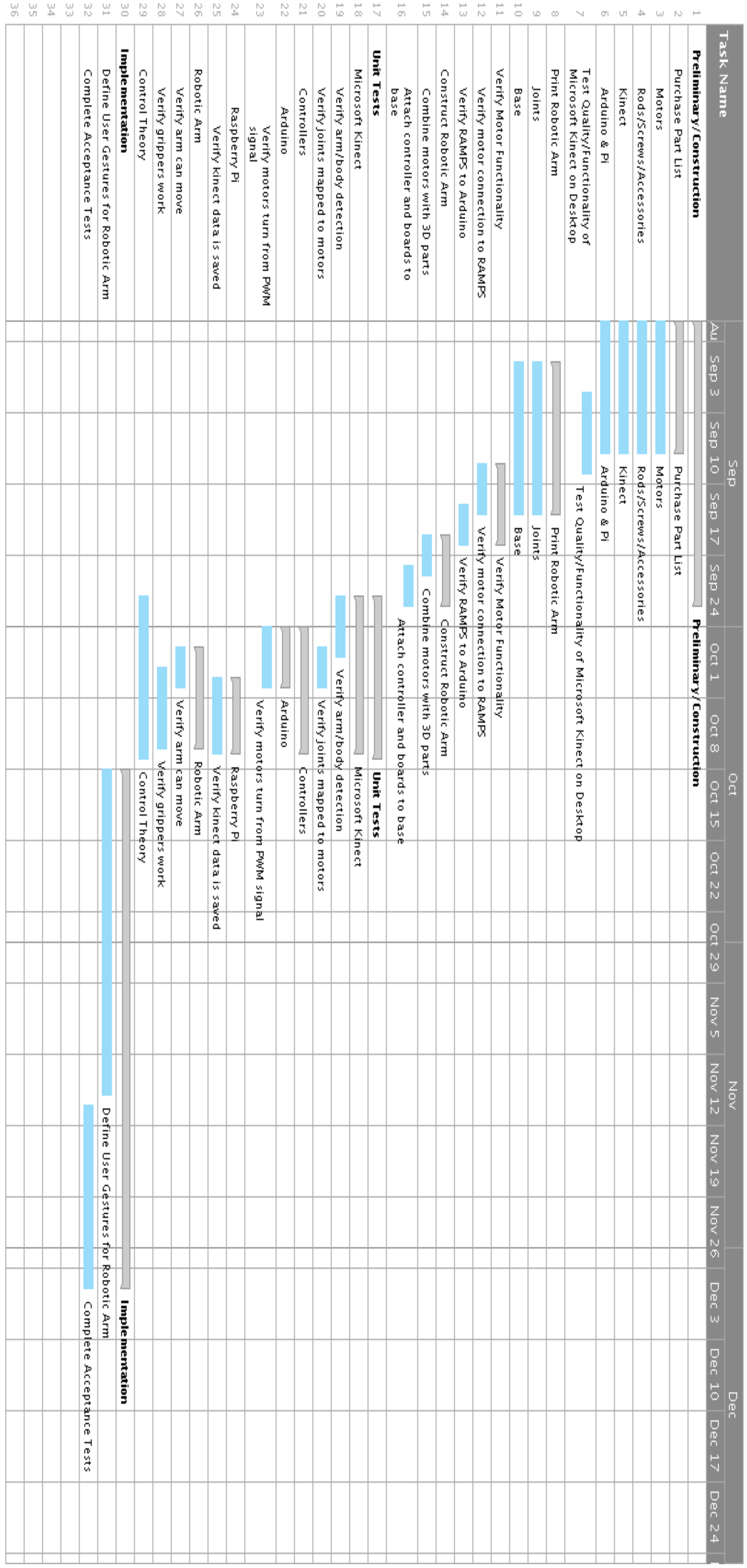

Figure 4. Gant Chart

10. Perspective

The final product met almost all of the requirements, but was not fluid and powerful as predicted. Looking back at the project, sourcing and building were predicted to be the easiest and quickest tasks in this project. However, that was not the case for original timeline for this task was two weeks, but ended up truly taking 5 weeks. If this task was to be done again, it would have been best to source all materials well in advance such as in the summer downtime. Furthermore, the building of the robotic was delayed significantly to due this sourcing issue, which caused a ripple effect throughout the project. However, once the robotic arm was built all next steps/tasks fell back into their predicated timelines. The only true deficiency from Project R.I.C.K was the inability for the claw to fully function in the sense of being able to carry heavy items such as a water bottle. This deficiency was not a hardware limitation but a software and timing limitation due to the code needed to perform this action would have takee additional time that was not available. The project was a great experience for it touched multiple fields of engineering from mechanical engineering to computer engineering. All in all, the project was a success for the majority of the requirements were fulfilled and on a personal note a team of complete strangers were able to not only work together but build a bond outside of this project.