Helping Hand: Intuitive, Tele-Operated Robotic Arm

Joshua E. Girard, CSE. Corey M. Ruderman, CSE. Daniel P. Travis, CSE. Jacob S. Wyner, CSE.

***Abstract*—As robotic technology becomes more prevalent in society, more intuitive control approaches will be necessary. This will make the technology more accessible to all users with little to no training. Helping Hand is a remotely controlled robotic arm using a novel human-robot interface. Helping Hand mimics the user's actions, resulting in an experience that is both precise and user friendly.**

I. Introduction

The applications and prevalence of robots in our society is expanding rapidly. Ever since the industrial revolution, man has sought to create machines that can accomplish tasks once only possible by a human. Originally, these machines were purpose-built, designed to perform only one or two highly specific tasks. However, the latter half of the previous century up until today have brought about more generalized applications. The human body is often regarded as the gold standard for general-purpose robots. This is not surprising as the human body is capable of performing a huge number of tasks. Therefore, by building a robot that is a human body analogue, or one that embodies certain elements, one can tap into this huge potential.

Of the entire human body, the most important limbs for manipulation are the arm and hand. In recent years, robotic arm technology has improved vastly and has been extended to many new areas. These range from construction site, to medical surgery robotic arms. The most famous example of the latter is the da Vinci surgical robot, which utilizes multiple extremely precise robotic arms to actually help perform a surgery. [7]

The challenges of these robotic arms are their remote-control systems; essentially, the way the operator interacts with and manipulates the robot. Currently, some of the most popular ways include using some kind of remote-control, joystick, or wearable device. While easy to implement, these methods remain complicated and unintuitive for the user. Our system can reduce—if not entirely eliminate—the amount of training needed to operate such systems.

In order to address these challenges, our team designed and constructed a remotely controlled robotic arm using a

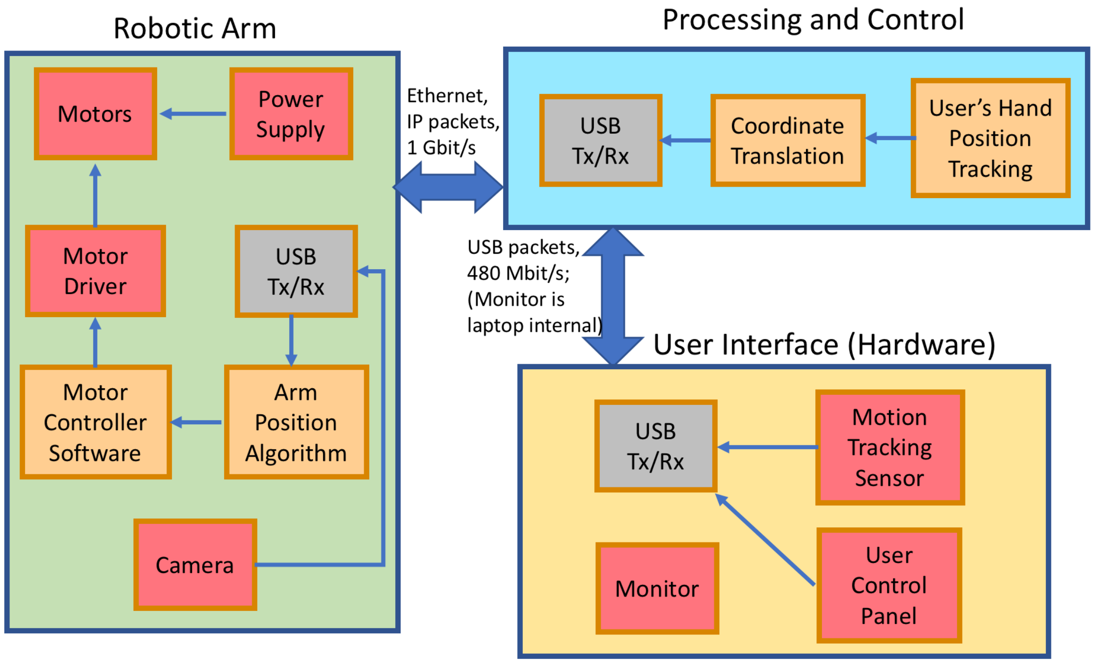


Fig. 1. This block diagram shows the organization of the Helping Hand project.

J. E. Girard (e-mail: jgirard@umass.edu).
C. M. Ruderman (e-mail: cruderman@umass.edu).
D. P. Travis (e-mail: dtravis@umass.edu).
J. S. Wyner (e-mail: jwyner@umass.edu).

novel human-robot interface, that is both precise and user friendly. The arm is mainly controlled by mimicking the user, meaning when the user moves their arm, the robot moves by a proportional amount.

Our system requirements are mainly due to the two large linear actuators that power to two main joints of the arm. Both motors draw up to 10A maximum [6] and therefore require a 20A power supply at a minimum just to power the arm. The microcontroller and the Raspberry Pi both require separate 1A power supplies. If the microcontroller does not have a separate power supply and is instead powered off the Raspberry Pi, the input can vary slightly which causes issues when reading analog voltages using the analog to digital converter (ADC). On the user interface side, all of the power is easily supplied by the user's laptop or desktop computer. Our system has been designed to meet the specifications as shown in Table I. The functionality test demonstrates the cohesiveness of system and all its specifications.

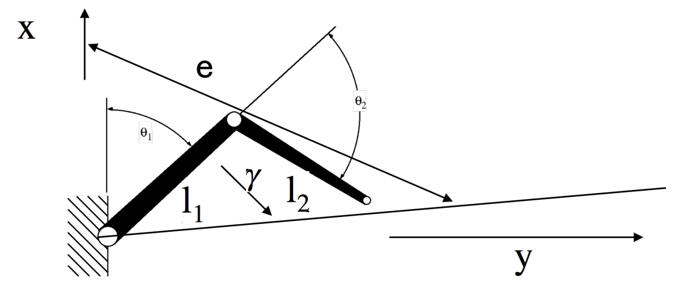
Because of the modularity of our system, the user interface side could in theory be very easily adapted to a larger or smaller system. However, the arm side is not as scalable. It is possible to change the dimensions slightly (changes of less than 10 cm in arm length), however, greater changes would require a complete redesign. In order for someone else to use our system they would merely need to turn the system on and place their hand over the sensor. Since the arm mimics the user, using the arm, even for a first-time user, is extremely simple and intuitive.

II. Design

A. Overview

The way in which our team has approached this problem consists of three main divisions: 1) the user interface, 2) processing and control and 3) the robotic arm hardware as shown in Figure 1. The processing and control portion of the system receives the three-dimensional coordinate of the user's current hand position from the user interface hardware. The hand position is determined using the Leap Motion controller and sent to the Raspberry Pi in the processing and control portion over Ethernet. The user control PCB also sends commands to the Raspberry Pi over the same Ethernet connection. We chose to use the Leap motion over similar technologies such as the Kinect [8] for several reasons. The Leap Motion is very accurate for hand movements, mainly because it was designed to do exactly that. In comparison the Kinect is designed to track the user's whole body. This results in decreased accuracy for reading just the hand positions and the state of being open or closed. The user control board provides the user with some control from the left hand to do things such as pausing and resuming tracking and emergency stop of the arm. These controls record using physical buttons rather than reading in the hand state with the Leap Motion.

The microcontroller receives its instructions from the processing and control portion of the system as shown in the block diagram. A Raspberry Pi model 3 is used to receive the coordinates from the user interface hardware portion of the system. The Raspberry Pi receives a three-dimensional coordinate of the user's current hand position from which it calculates the joint angles needed to reach that point. These calculations are done using inverse kinematics equations. See Figure 2.



$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$l = \sqrt{x^2 + y^2}$$
$$l_2^2 = l_1^2 + l^2 - 2 l_1 l \cos\gamma$$
$$\Rightarrow \gamma = \arccos\left(\frac{l^2 + l_1^2 - l_2^2}{2 l_1 l}\right)$$
$$\frac{y}{x} = \tan\varepsilon \quad \Rightarrow \quad \theta_1 = \arctan\frac{y}{x} - \gamma$$
$$\theta_2 = \arctan\left(\frac{y - l_1 \sin\theta}{x - l_1 \cos\theta_1}\right) - \theta_1$$

Figure 2. Arm joint configuration and inverse kinematics equations.

The inverse kinematics algorithm shown above works well for manipulation in 2D. However, in order to add support for 3D, it was adapted to work by first calculating the necessary base angle, then performing a mathematical rotational transformation using that base angle. This allows the shoulder and elbow joint angles to be calculated in the 2D plane facing the target point.

The calculated angles are then sent to the microcontroller. The Raspberry Pi was selected for this task because of its ability to perform the inverse kinematics equations quickly and reliably, its ability to communicate over Ethernet with the user interface hardware, and its ability to interface a webcam with it. The Raspberry Pi also allows us to add additional computations and tasks as needed without significantly affecting the performance. This portion of the system meets the specifications of latency and movement criteria as shown in Table I.

The robotic arm system includes both the mechanical design of the arm and the electronic hardware used to control the arm. The control of the arm is implemented using an Atmel microcontroller which commands each joint angle. Using this angle, the microcontroller calculates the necessary speed to set the joint motors. This calculation is done using the desired angle and the current angle as inputs to a basic feedback system which outputs a speed and direction for each motor. See Figure 3.

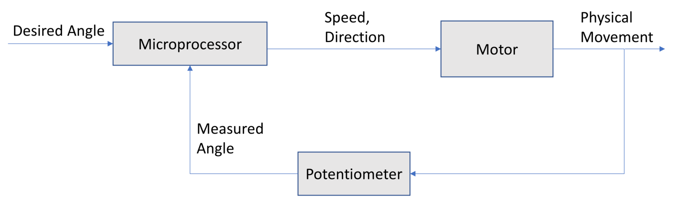


Figure 3. Control Feedback Loop.

The base motor implements a Lead Compensator feedback control system. See Figure 4. The purpose of this is to ensure that the rotational movement is very fluid and precise, as well as not overshooting the target angle. This is critical since there is a large amount of rotational momentum when the base is turning at speed. In Figure 4, the u terms are the position, e terms are the error, T term is the discrete time step interval, and K is the tuning parameter. The a and b parameters are set such that a<alpha<b, where alpha is the inverse of the system response time, or system time constant. The u_{k+1} is the calculated position, which determines speed and direction.

$$\frac{u_{k+1}-u_k}{T}+bu_k=K[\frac{e_{k+1}-e_k}{T}+ae_k]$$

$$\Rightarrow u_{k+1}=T(K[\frac{e_{k+1}-e_k}{T}+ae_k]-bu_k)+u_k$$

Figure 4. Lead Compensator.

The speed and direction are then given as the input to the H-bridge which actuates the motors, and the base servo. The Atmel microcontroller was chosen due to both the reliability and robustness of operation and its simplicity in implementation. See Figure 5. The H-bridge was chosen simply based on the current and voltage specifications for the motors.

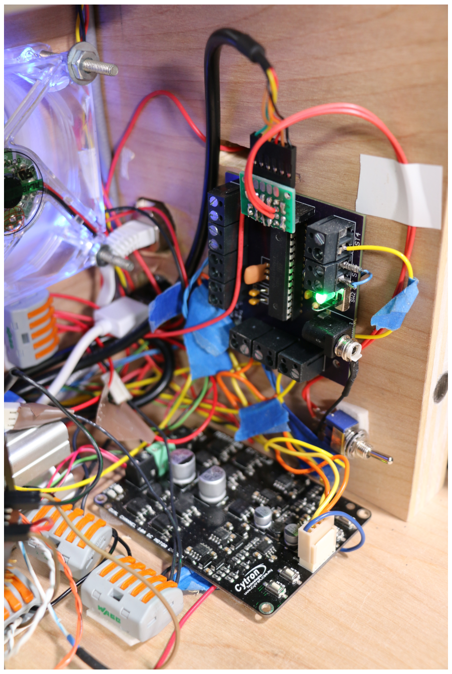


Figure 5. H-bridge (below) and microprocessor (above).

The physical arm was designed from the ground up. The arm is essentially built around two linear actuators which can support up to 110 pounds while in motion [6]. These two motors provide the two degrees of freedom in the vertical plane, while a base motor provides the degree of freedom in the horizontal plane. The physical structure of the arm is constructed of aluminum which provides us with significant strength while keeping the weight to a minimum. The arm system will meet the specifications of minimum range of motion, minimum speed, latency, and movement criteria as shown in Table I.

Requirement	**Specification**
Lifting Strength	Should be able to lift at least 1 lb.
Range of Motion	The robot should be able to reach every point in a workspace 2' wide, 2' deep, and 1.5' tall.
Latency	Robot should move within 250ms of the user moving.
Speed	The movement of the robot should be at least 5 inches per second.
Functionality	User should be able to move five rocks from the workspace into a bowl in under 5 minutes.

Table I: List of system requirements and specifications.

B. *Hand Position and Motion Tracking Sensor*

These sub-blocks of the Processing and Control block and User Interface block are concerned with tracking the user's hand position in three dimensions and in real-time. The motion tracking sensor will be placed on the table in front of the user. The sensor we are currently using is the Leap Motion Controller [1]. The Leap Motion Controller LM-010 is a 3x8cm area package which consists of three IR emitters and two IR cameras and is interfaced with a computer via USB 2.0 [2]. This sensor has a tracking accuracy of less than 0.2mm for stationary positions and 1.2mm for dynamic ones [2]. It has a latency of 5ms when in High-Speed mode, and 12ms when in Precision mode [3]. It's effective tracking area is 25mm to 600mm above the sensor, with a field-of-view of 150 degrees spanning radially upwards from the sensor [1]. From our own testing, the sensor typically samples at 100Hz, although this can vary by up to 5Hz depending on CPU usage. However, conditions such as lighting have a quantifiable impact of performance. Moreover, a calibration process is necessary to achieve the optimal accuracy for a particular environment, although un-calibrated performance is usually sufficient for our application.

The experiment we conducted to measure the effective tracking area of the sensor was be to move our hand back and forth, up and down, side to side over the sensor, creating a point-cloud of tracked Euclidian (xyz) positions. Then taking this point-cloud, we generated the convex hull. This convex hull is the definitive volume within which hand tracking is possible. This experiment was then repeated under a different set of lighting conditions to see if that had an effect. See Figure 6. The results of this experiment demonstrated that the Leap has a tracking area of 43, 37, and 37.5 centimeters in the X, Y, and Z axes respectively. Since this is not as large as the workspace specification, a scaling is applied to map small movements of the hand to larger movements of the robot. Specifically, we are using scaling factors of 3, 2.5, and 1.2 in the X, Y, and Z dimensions respectively. These factors were chosen such that the full range of motion of the Leap is mapped to the full range of motion of the robot in each axis. Since the robot has more mobility in some directions than others, each axis is scaled differently accordingly so that the robot's full range of motion is accessible.

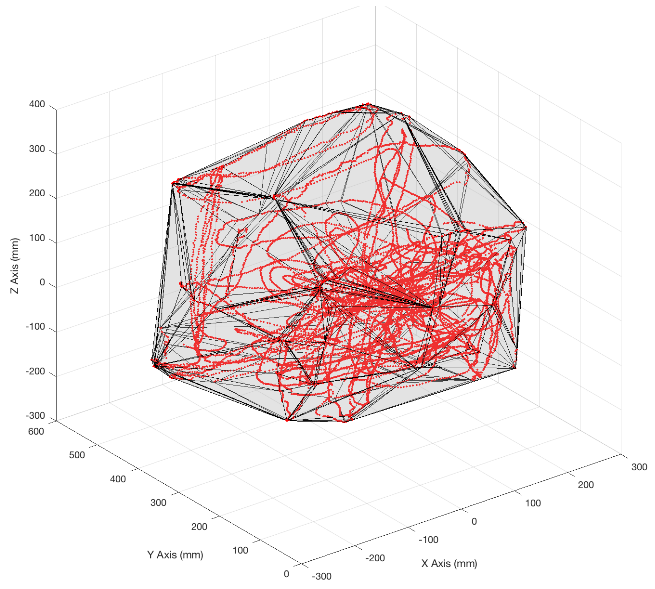


Figure 6. Leap Motion tracking volume.

The motion tracking system and processing, from sensor to Euclidian points, is handled automatically through the Leap Motion's driver, provided as part of the Leap Motion's software package. Since the sensor samples at 100Hz but our system only transmits data between the user's computer and the robot's embedded system at 20Hz, we have more data than can be sent. We currently only take every 5th sample and transmit it.

For the sake of monotonous tasks, we have implemented a replay feature. This allows the user to record their actions and loop back over them for however long they'd like. The protocol for sending data to the Raspberry Pi made implementing this feature straightforward; essentially, all we needed to do was save these data points to a .csv file and loop through the file each time we wanted to replay them. This feature is especially useful for automated processes, freeing the user from repeating the same task more than once and avoiding issues like muscle fatigue after prolonged use.

C. Robotic Arm

The Robotic Arm block encompasses the major hardware component of the system. See Figure 7. The arm is constructed with four main pieces, the base, shoulder, elbow, and gripper. The shoulder and elbow stages sit on top of the base, which uses a turntable bearing to provide 270 degrees of rotation along a single plane, in front of the robot. The base is controlled by a stepper motor, with a machined aluminum adapter connecting it to a 25:1 gearbox. The gearbox-motor system is then mounted on two sliding rails, to adjust its distance from the rotation shaft. The base motor is connected to the shaft with a sprocket and chain system. The chain can be tensioned with a screw, which is accessible from the outside of the box. The shoulder and elbow joints are both powered by linear actuators, allowing for controlled, stable motion in a single plane. By combining the capabilities of all three joints, we are able to achieve a working area of approximately 2ft x 2ft x 1.5ft directly in front of the robot.

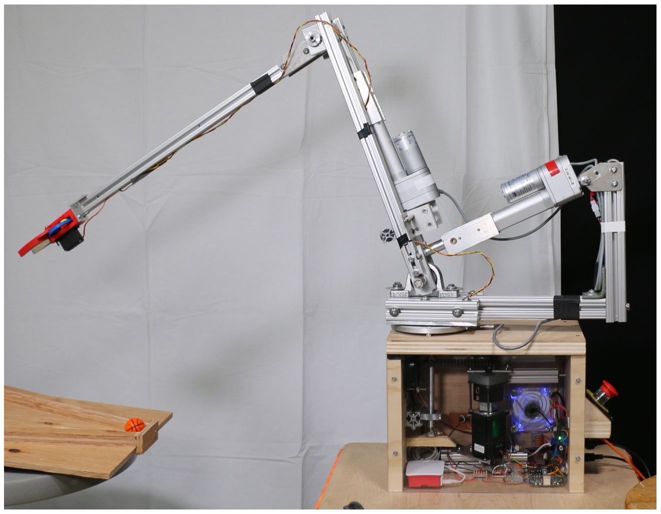


Figure 7. Helping Hand Prototype

The frame of the arm is built primarily from 1 inch 8020 T-slotted aluminum extrusion. This particular stock was chosen for its combination of strength and being lightweight. Alongside the aluminum stock, there are rotary bearings on the two upper joints to provide a smooth rotation of the joints. The entire arm is mounted on an enclosure built from 1in plywood, which provides plenty of strength to support the whole system.

To test the functionality of the arm, a manual control unit was built. The manual control unit delivers 12V in either direction to the motors from a momentary switch. This allowed for testing the arms total workspace, and for further developing the physical components of the arm.

The final phase of arm development was the gripper. The gripper is needed to pick up objects, and in our case, ping pong ball sized rocks, foam basketballs, and small blocks of wood. We chose to use the open source, 3-D printable Mantis Gripper (see Figure 8). [9] This gripper was chosen for its combination of strength, size, and cost-effectiveness. By being able to 3-D print the majority of the parts, we will be able to make the best use of our budget.

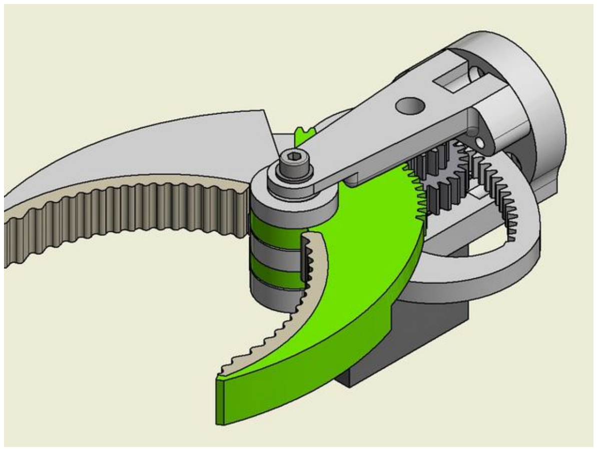


Figure 8. Mantis Gripper 3D Render

D. User Control Board

The User Control board provides additional functionality for the user, which would be difficult to control with hand gestures alone. The board is designed to be controlled with the user's non-dominant hand. The board features two switches and a button, which are connected to our custom PCB featuring an ATMega328P [10]. The button allows the user to stop the system in case of emergency, and the switches all them to pause and resume, as well as reset the system. The user control board is connected to the user's computer via USB. See Figure 9. (Note: The center knob is unimplemented for future expansion.)

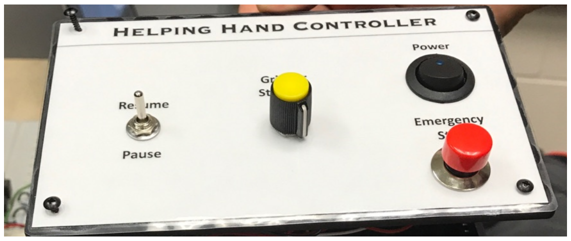


Figure 9. User Control Board/PCB.

III. Project Management

Our team showed a lot of vitality and perseverance since the beginning of this project, and throughout that we continued to learn how to work together efficiently and effectively. With effective communication and personal as well as frequent meetings and clearly delegated tasks, we were able to accomplish all of our specifications as outlined in Table I. As promised, we built and designed a robotic arm with precise positioning in 3 degrees of freedom. Since our midterm report we have integrated horizontal movement into the systems, which included making many changes to the control algorithms, changes on the user side software, as well as improvements to previous control algorithms. Lastly, we completed and integrated the user control panel.

Each member of the Helping Hand team has contributed important and specific aspects to the overall project and has a specific expertise. Corey Ruderman's expertise lies in software, specifically interfacing it with hardware. Corey has designed the protocol for intersystem communication, assisted with the arm control algorithms, and worked on the team website. Daniel Travis has experience building mechanical systems and designing printed circuit boards. Daniel has researched the most optimal design for our arm, and constructed the physical arm, as well as the user control panel prototype. Jacob Wyner is skilled at electronic hardware integration and embedded system programming and design. Jacob implemented the electronic arm hardware and took the lead for developing the arm control algorithms. Joshua Girard is proficient in programming, networking, and embedded systems. Joshua developed motion tracking algorithms for both the Leap Motion Controller and the Kinect Sensor as well as improved feedback control of the base motor.

Although each team member had their own contributions to the project, everyone helped out to make sure that the goals that had been set were met. When it came to designing the physical arm and the protocol for intersystem communication, everyone helped to come up with an optimal solution. Whether this meant helping to troubleshoot or choosing materials for the arm design, the team was there to help when needed. Overall, it was a group process to meet all of our FPR deliverables and system specifications.

To complete the various aspects of the project, the team remained in constant contact with each other via GroupMe. We continually updated each other on individual progress and met once a week with our advisor to touch base. This meeting gave us a chance to ask questions when problems arose and get invaluable guidance. Also, we reflected on the progress that had been made and where we planned to go next. We managed to stay on the same page through these meetings as well as email contact.

IV. Conclusion

Project Helping hand has completed our proposed project on schedule. We have accomplished all of our proposed specs (aside from a slight deficiency in one degree of freedom on our speed spec in Table I) and we have produced a full working prototype. If we had more time we would have worked on improving physical robustness, for example perhaps by machining our gripper rather than 3D printing it, and we would have liked to add a 3D camera to the arm so that it could be controlled remotely without the operator losing their depth perception. Despite the fact that in our final demo with our evaluators we realized that the arm could not move on a line parallel to the floor at the required speed, we agreed with our evaluators that the speed was fairly arbitrary to begin with and the slowness did not affect the overall functionality of our arm.

Overall our project was a success, we produced a fully functioning prototype very similar to our initial idea we formed back in September. We are very pleased with the way it turned out and we hope that future groups can utilize our project for their own unique ideas.

Appendix

Below is the development and estimated production costs of the system.

Part	Development	Production (1000)
Gripper System	$50	$20
Stepper Motor	$158	$100
Linear Actuator (2)	$260	$160
Servo Motor	$5	$2
Motor Driver	$28	$12
Custom PCB (2)	$60	$10
Raspberry Pi	$35	$20
Leap Motion	$60	$40
Webcam	$35	$15
Power Supply	$30	$10
8020 Aluminum (10ft)	$35	$10
Mounting Hardware	$30	$5
Total	$786	$404

Acknowledgment

Our group would like to thank our advisor, Professor Duarte, for giving us such great insight and advice throughout our project. A special thanks to Professor Hollot for his invaluable control systems knowledge. We would also like to thank Professor Frasier, Fran Caron and the staff at M5 for providing us with supplies.

References

[1] Leap Motion, Inc. API Overview. [Online]. Available: https://developer.leapmotion.com/documentation/v2/cpp/devguide/Leap_Overview.html. [Accessed Web. 12 Nov. 2017.]

[2] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler. (2013). Analysis of the Accuracy and Robustness of the Leap Motion Controller. Sensors (Basel, Switzerland), 13(5), 6380–6393. http://doi.org/10.3390/s130506380

[3] R. Bedikian. (2013, July 13) *Understanding Latency* [online]. Available: http://blog.leapmotion.com/understanding-latency-part-2/ [Accessed Web. 12 Nov. 2017.]

[4] H. Jin, Q. Chen, Z. Chen, Y. Hu, J. Zhang, Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task, In CAAI Transactions on Intelligence Technology, Volume 1, Issue 1, 2016, Pages 104-113, ISSN 2468-2322, https://doi.org/10.1016/j.trit.2016.03.010. (http://www.sciencedirect.com/science/article/pii/S2468232216000111)

[5] 8020 Inc. "The Core Building Blocks." *80/20® Inc.*, 2017, 8020.net/university-tslot.

[6] GlideForce. "Linear Actuator Data Sheet." Concentric International. GlideForce. https://www.pololu.com/file/0J1238/LD_Linear_Actuator_Data_Sheet_1607.pdf

[7] B. Davies, "Robotic Surgery – A Personal View of the Past, Present and Future," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 54, 2015.

[8] Developing with Kinect for Windows [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/develop

[9] Hölldorfer, Andreas. "Mantis Robot Arm - Part 1 - The Gripper." *ChaozLabs*, 1 Jan. 2017, chaozlabs.blogspot.com/2016/04/mantis-robot-arm-part-1-gripper.html.

[10] ATmega328/P Datasheet. Atmel, 2016. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf