Spring 2019

IR Motion Tracking Robotic Arm

Gavin Low
lowg19@up.edu

Andrew Doan
doana19@up.edu

Avery Guillermo
Guillera19@up.edu

Dayna Yoshimura
yoshimur18@up.edu

IR Motion Tracking Robotic Arm

Final Report

Team Members:
Spring Team Lead: Andrew Doan
Spring Documents Coordinator: Avery Guillermo
Spring Presentations Coordinator: Dayna Yoshimura
Spring Webmaster: Gavin Low

Team Website: https://engineering.projects.up.edu/lowg19/

Faculty Advisor:
Dr. Peter Osterberg

Industry Advisor:
Dr. David Laning

University of Portland
School of Engineering
5000 N Willamette Blvd.
Portland, OR 97203-5798

Phone: 503 943 7292
Fax: 503 943 7316

Revision History

Table 1 - Revision History

Rev	Date	Author	Reason for Change
0.90	15 March 2019	Avery Guillermo	First draft released
0.95	29 March 2019	Avery Guillermo	Second draft released
1.0	12 April 2019	Avery Guillermo	Final draft released

Table of Contents

Table of Figures

Table of Tables

Introduction

Robotic arms are used in multiple areas of day to day life such as in the industrial, agricultural, and medical fields. Typically, robotic arms are controlled with remote controls or through automated programs. The IR Motion Tracking Robotic Arm is a senior design project that explores alternative robotic arm control methods. With the use of a LEAP Motion Controller, the location of the user's physical forearm and fingers are tracked. This data is used to send signals to an Arduino Uno which in turn powers the servos on the robotic arm. The final product is a pick and place robotic arm that is capable of picking up a wooden block and placing it on a new spot; it is fully controlled by tracking the user's own arm movements.

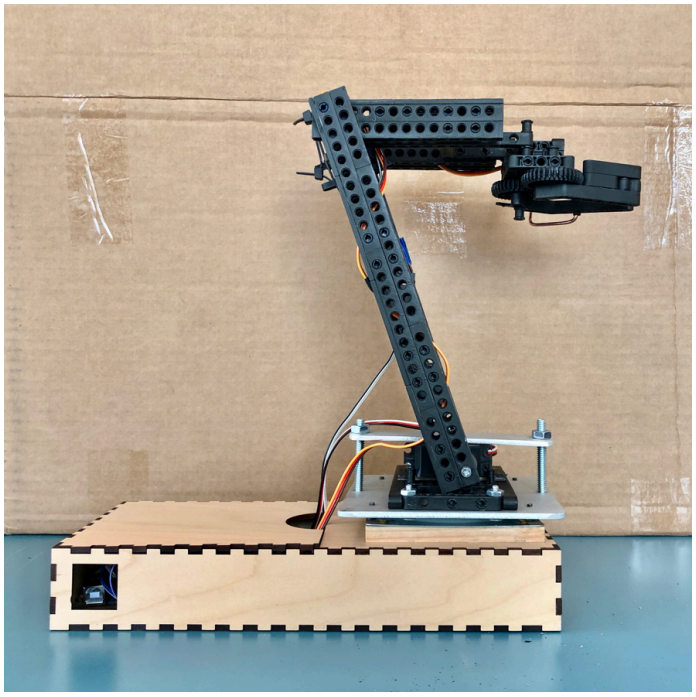

Figure 1 - IR Motion Tracking Robotic Arm

Figure 1 is an image of the final product. The robotic arm has four degrees of freedom — a rotating base, an elbow, a wrist, and a claw. The robotic arm is housed inside a laser cut wooden frame. This frame also contains the electrical components needed for the robotic arm to run, creating a sleek and portable product. The only wires coming out of the frame are the USB wire to connect a PC to the Arduino Uno and the power and ground leads of a 5V power supply, which is used to power the servos.

The robotic arm itself is built out of LEGOs. The choice to use LEGOs instead of a material such as aluminum came as a recommendation from Industry Advisor, Dr. David Laning. Since the team had limited mechanical design experience, Dr. Laning suggested LEGOs for ease of building and testing a prototype. Since the purpose of this project is to demonstrate a proof of concept and is an education tool, LEGOs were an acceptable choice to proceed with. With more funding and experience, the team could replace the mechanical structure towards more robust, sturdy materials.

Additionally, the team decided to split the year into two semester long goals. The goal of the fall semester was to create a complete mechanical design and the goal in the spring semester was to implement the LEAP Motion Controller. Since the team worked very well together, they achieved each of those goals and created a final product that was very close to what was originally envisioned. The remainder of this document will provide a more detailed look at the technical portion of the project, the milestones, the risks, and a conclusion.

Technical Outcomes

High-Level Architecture

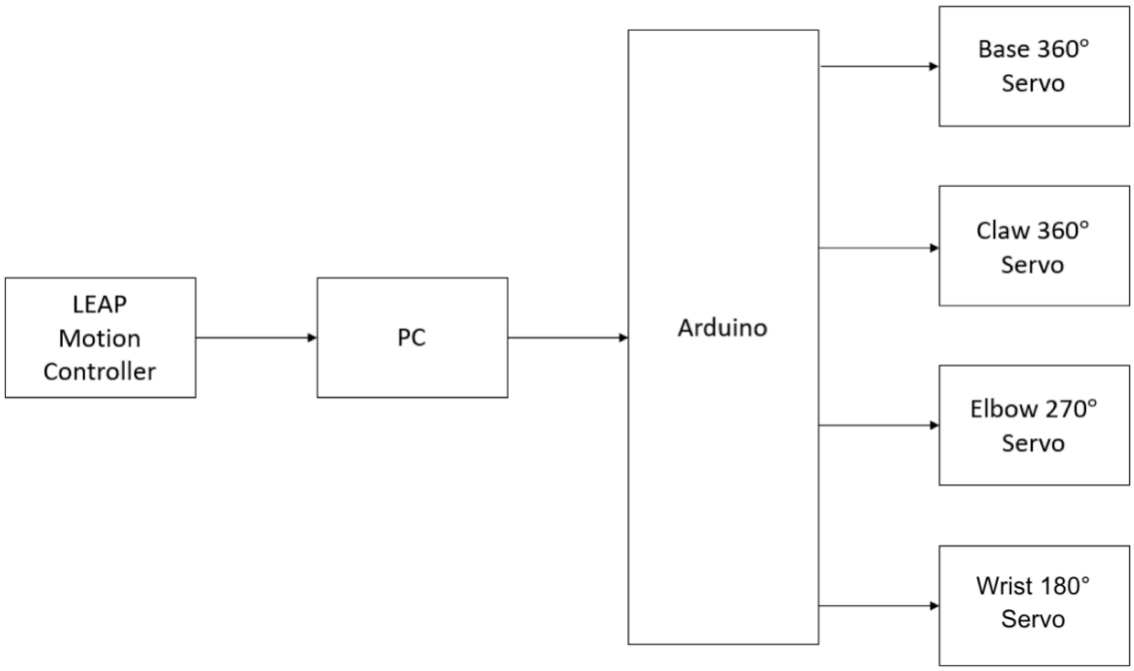

Figure 2 - High Level Design Of The IR Motion Tracking Robotic Arm

The IR Motion Tracking Robotic Arm is comprised of seven major components. Only one of these components (LEAP Motion Controller) is for user interface. The remaining six components (PC, Arduino, and the four servos) are required for data interpretation and control of the robotic arm. Of the seven components, the LEAP Motion Controller and the Arduino play major roles in the function of the device. The information within the high-level design shown in Figure 2 is collected and processed in real time.

The LEAP Motion Controller utilizes an IR camera to record the user's hand movements and exports the captured frames to a PC. The minimum system requirements of the PC are that it must be operating on Windows 7+ or Mac OS X 7+ and contain the Leap Motion Software. Within the PC, the frames are interpreted with respect to a coordinate plane integrated in the LEAP Motion Controller's software. A nodeJS (Javascript runtime environment) script will import the coordinates of the user's palm and the tips of the user's thumb and index finger and prepare the information to be distributed to the respective servo motors. The processed information serve as an input to the Arduino component. This component uses the input information to control the servo motors in the robotic arm. The palm's X coordinates from the LEAP data are utilized by the base servo, the palm's Y coordinates are used by the elbow servo, and the palm's Z coordinates are utilized by the wrist servo. The finger tip coordinates are used by the claw servo. Ultimately, the Arduino distributes the data received to the respective servo, thus allowing the servos to mimic the user's movement.

User Interface

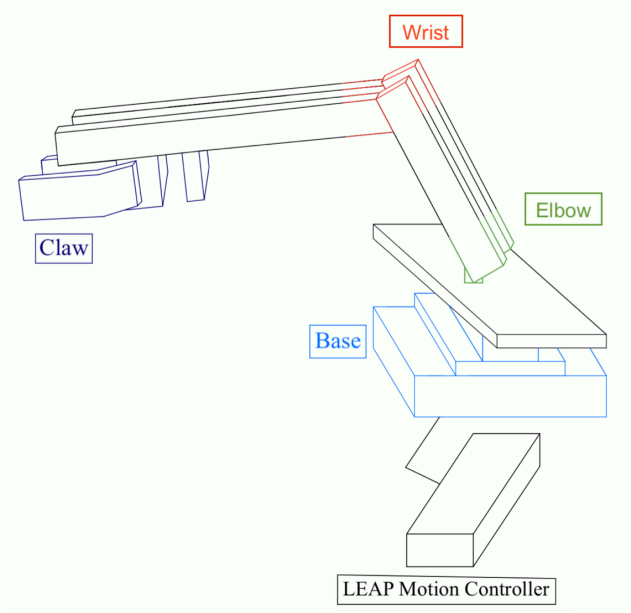

Figure 3 - IR Motion Tracking Robotic Arm Design

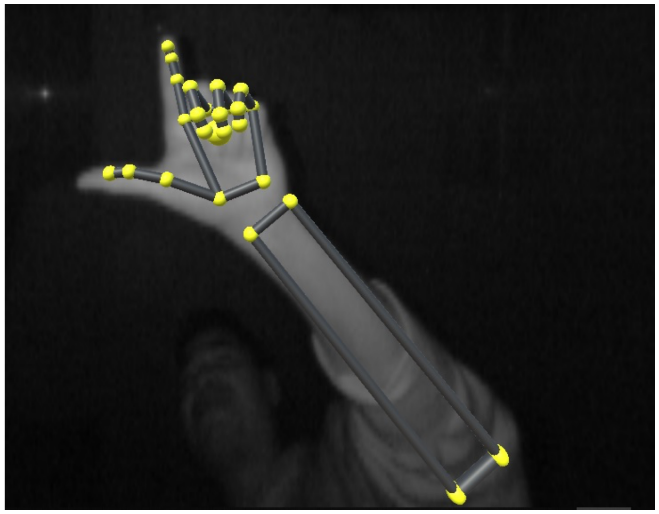

Figure 4 - Vectors And Points Generated By The LEAP Motion Controller

The IR Motion Tracking Robotic Arm allows the user to control the arm without the use of switches or potentiometers. While the user moves their arm in front of the IR camera, as shown in Figure 3 and Figure 4, information is recorded and relayed to the robotic arm's internal components. This process produces movements of the robotic arm that mimic the user's movements in real time.

1. LEAP Motion Controller
   - Observes the user's arm movement and translates the images to position values based on its internal coordinate system. Using the position values generated, the LEAP Motion Controller recreates the user's hand in a digital space as shown in Figure 4. Positions and vectors are recorded for the hand, finger tips, and forearm.

Mechanical Components

The final IR Motion Tracking Robotic Arm assembly, pictured in Figure 5, consists of five main mechanical components: a rotating base, a forearm, a wrist, a claw, and housing. The forearm, wrist, and claw are built out of LEGO Mindstorms parts, while the rotating base and housing are made out of wood and aluminum. The LEGO pieces were superglued together and painted matte black to create a sleek and sturdy final product. As mentioned earlier, the choice to use LEGOs came from Dr. David Laning, the team's industry advisor. The LEGOs allowed the team to build a working design relatively quickly, providing more time for testing and implementing the LEAP Motion Controller. The remaining portion of this section discusses each component of the robotic arm. Refer to the Design Document for a more in-depth discussion of each component. Figure 5 is an annotated picture of the robotic arm and labels each of the five components.

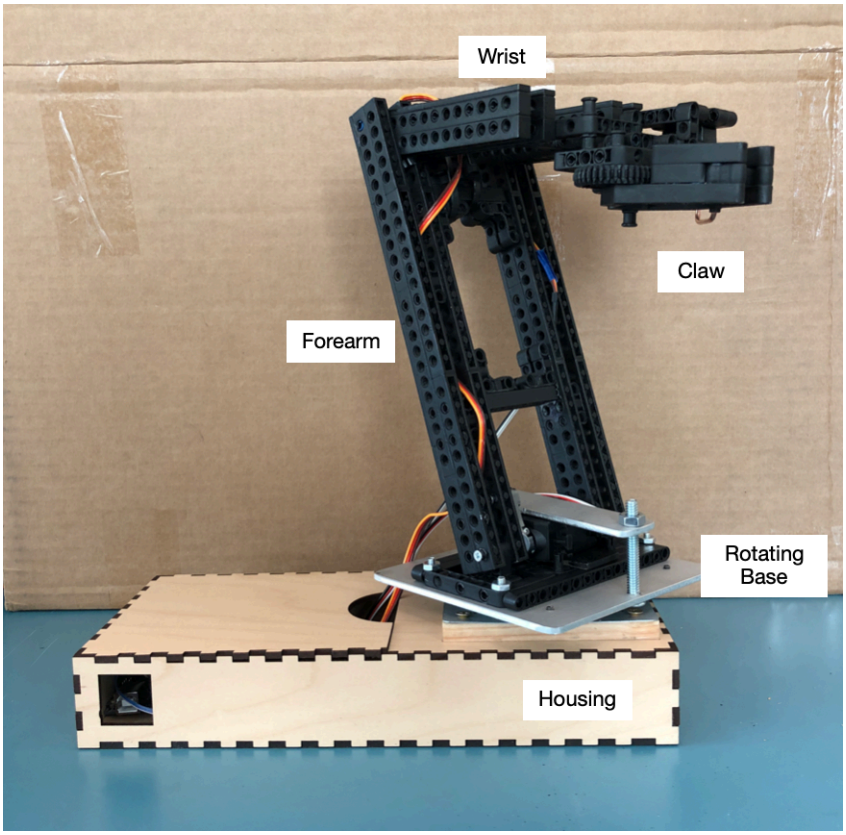

Figure 5 - Motion Tracking Robotic Arm (Isometric View)

The Base

The rotating base is the first mechanical component in the robotic arm. The base consists of an aluminum turn table mounted onto two wooden blocks that serve as legs. A 270 degree servo is fixed between the two wooden legs to allow the base to rotate left and right. A large aluminum plate is attached to the turn table to provide a space to mount the robotic arm. Figure 6 below is an isometric view of the rotating base.

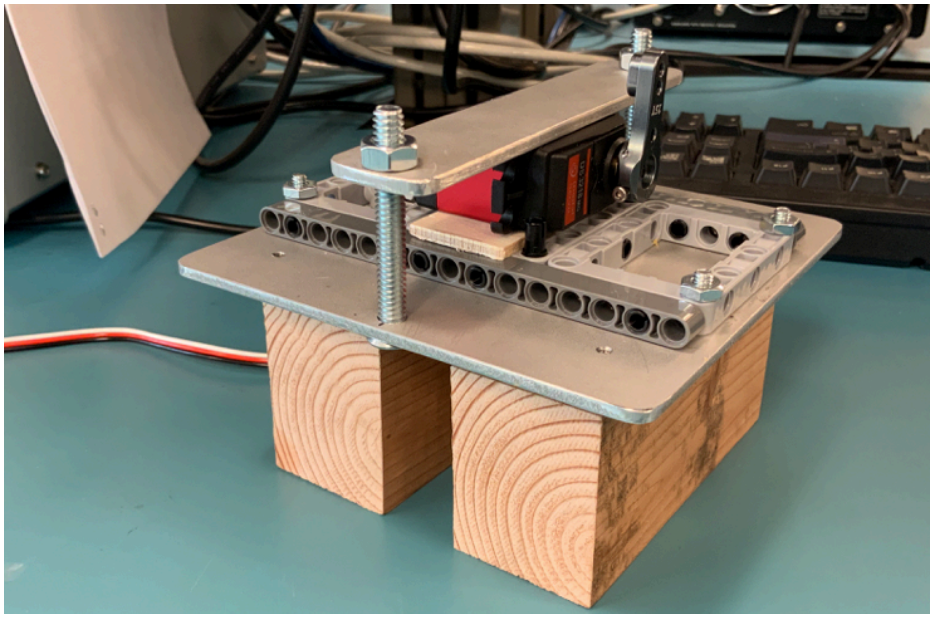

Figure 6 - Isometric View Of The Rotating Base

The rotating base was manufactured with the help of Jacob Amos, a lab technician at the Shiley School of Engineering. A LEGO plate is then anchored on top of the aluminum plate and is used to mount the forearm. The rotating base also supports the forearm servo, as shown in Figure 6. This servo is securely held in place by an aluminum bar.

The Forearm, Wrist, and Claw

The forearm, wrist, and claw make up the rest of the robotic arm. Each of these components are made out of LEGO Mindstorms bricks. LEGO H-shaped connectors and beams provide additional reinforcement and prevent twisting. Figure 7 provides a top view of the robotic arm.

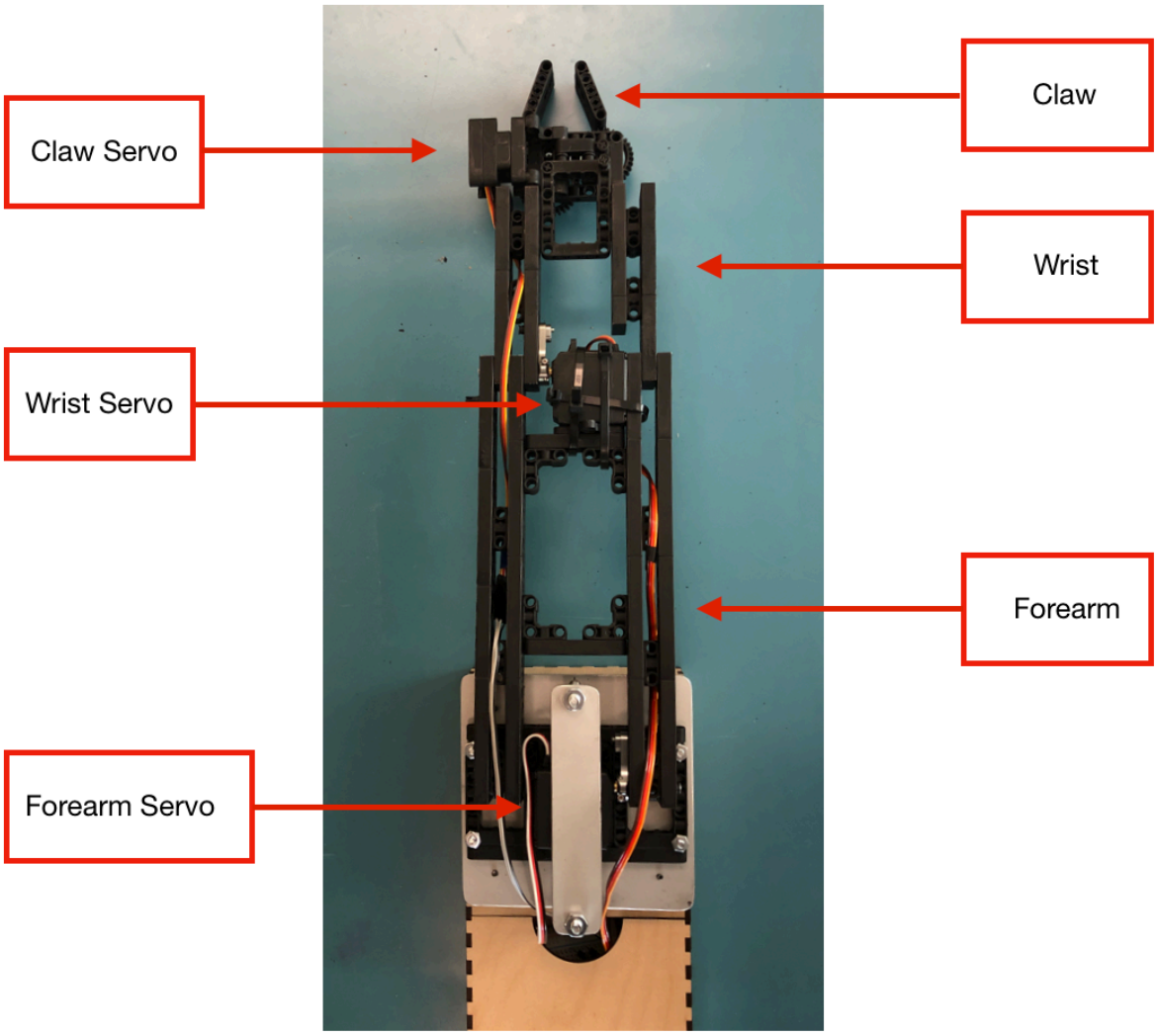

Figure 7 - The Forearm, Wrist, And Claw

Figure 7 is a top view of the robotic arm and labels a few key components. The first component is the forearm. It is connected to the LEGO plate mentioned in the rotating base section. The forearm servo, which is mounted underneath the aluminum bar, is a 180 degree servo that allows the robotic arm to move up and down. The next component is the wrist. It is connected to the forearm where the wrist servo is located. This is a 270 degree servo that allows the robotic arm to move inward and outward. The servo is tightly zip tied to the side of the forearm to keep it stationary. The last component is the claw. The claw is what allows the robotic arm to pick up and place the wooden block. The claw servo is a 180 degree servo and is super glued to the claw. When the servo rotates, it pushes and pulls a copper bar, which in turn opens and closes the claw. The wooden block is pictured below in Figure 8. It weighs about half an ounce and is about 1.5 cubic inches in size. The maximum load this robotic arm can handle is 5 ounces, or 142 grams.

Figure 8 - The Wooden Block

Housing

The final component of the robotic arm is the wooden housing. This is the only new component since the design document. The idea of a housing came late in the design process as a way to make the final product sleek and portable. It is made out of wood that has been laser cut in the Maker Space in the Shiley School of Engineering. There are two sections to the housing: the main box and the lid. The main box holds the robotic arm in the front and provides space for the Arduino Uno and breadboard in the back. The removable lid provides easy access to the internal electronics as well as acting as a protective cover while the arm is in motion. A small hole on the back side wall serves as a place to run the power brick leads as well as the Arduino USB cable into the housing. Figure 9 below is a picture of the housing with the lid removed.

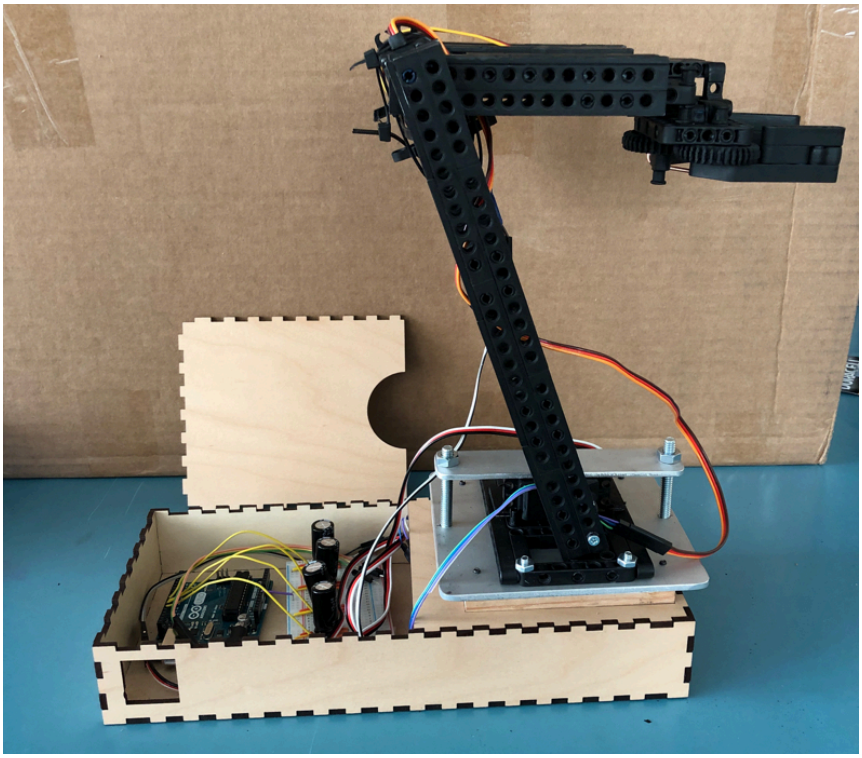

Figure 9 - Housing

Electrical Components

During the fall semester, as the team built more refined mechanical prototypes, the team needed a way to fully test the limits of each extremity. The team accomplished this through potentiometers. From this, the team could specify accurate movements to control each individual servo as shown in Figure 10. In addition, the connections to the Arduino, Power, and Ground are neatly shown in Figure 11.

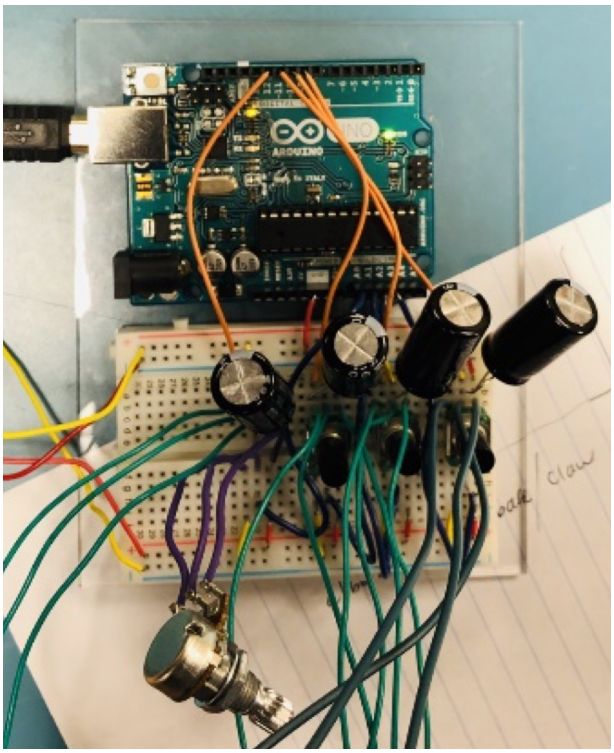

Figure 10 - Arduino Setup With Potentiometers

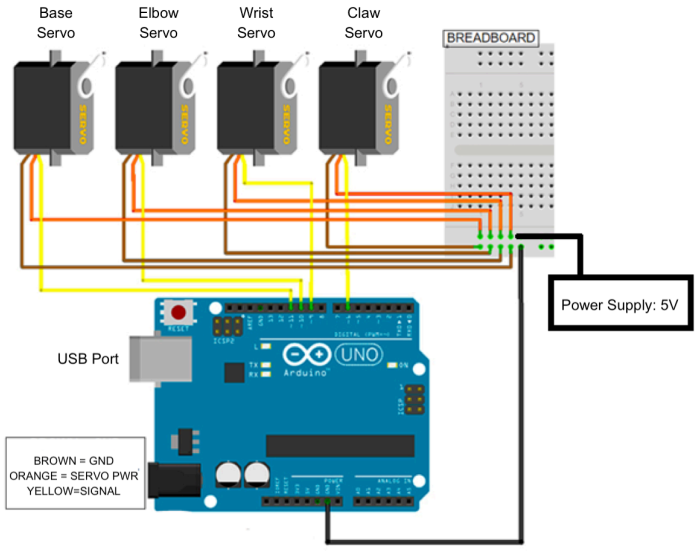

Figure 11 - Electrical Schematic

The Arduino would process the variable resistances of the potentiometers and translate it into servo angles; these servos were powered by a DC Power Supply. However, the voltage of the power supply would slightly fluctuate, which caused the servos to jitter because it would adjust for every tiny voltage change. The team then implemented large bypass capacitors to smooth out the incoming voltage to reduce the shaking, as show in Figure 10.

Once the LEAP Motion Controller was implemented in the spring, there were small improvements to the electrical components. The servos now draw power from a 5V, 20A power brick as shown in Figure 11. During full load, the total current drawn from all four servos is 12 Amps. This is well below the power brick's rating of 20 Amps. From this, the total power consumed during full load conditions is 60 Watts. Another improvement was the use of ribbon wires to consolidate the servo wires to create a cleaner circuit since the potentiometers are no longer needed.

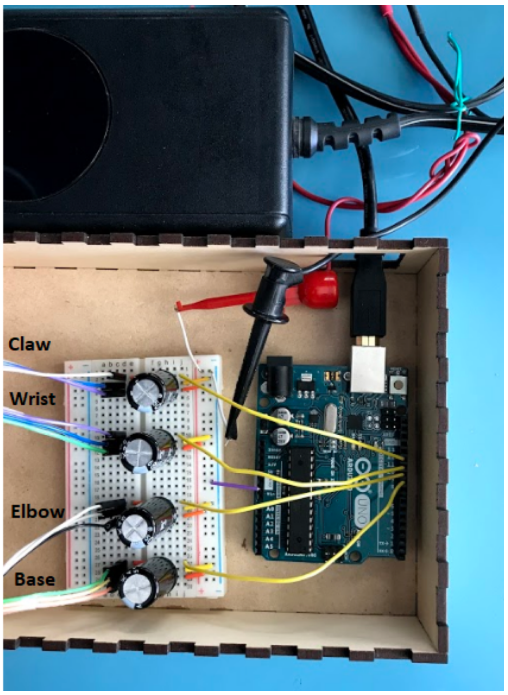

Figure 12 - Final Arduino Setup Without Potentiometers

Software Components

The data handling for this project is done through a PC and an Arduino Uno. The team decided to use an Arduino microcontroller because the Arduino can easily translate the data from the LEAP Motion Controller to control the robotic arm. In addition, the Arduino includes vast firmata libraries, which are generic protocols for using software from a host computer to communicate with the microcontroller. This is helpful because JavaScript is the language primarily used by the LEAP Motion Controller; in addition, the Johnny-Five package is the JavaScript robotics programming framework. These programs allowed the team to interface the LEAP Motion Controller with the Arduino. By creating various functions and calculations through coding, the team's program is able to take in real-time data from the LEAP, process that

information into servo angles, and then give that data to the Arduino to control the servos. The complex calculations involved are fully described in the Design Document. In addition, the entire code used to interface the LEAP Motion Controller with the Arduino is shown in Appendix A. Additional features the team implemented is a startup and sleep function. This feature sets the robotic arm to be fixed at a certain position when the LEAP Motion Controller is not tracking any user arm data. On top of this, the team added restrictions on the servo angles to prevent the arm from collapsing on itself.

Performance

The final performance of the robotic arm resulted in a critically-damped motion in the servos that properly mimic the user's movements. During the testing process, the team decided to implement certain adjustments to improve the user interface as well as provide a smoother movement of the robotic arm. For instance, during the Mechanical Robotic Arm Tests, all servos were able to surpass the minimum angles. Thus, the team determined that setting bounds on the angles was needed to prevent the user from moving in such a way that would collapse the robotic arm. In addition, the team implemented a sleep function that would prompt a smooth movement to a resting position when there is no user input. Given the Mechanical Robotic Arm Tests, LEAP Motion Controller Tests, and Arduino Microcontroller Tests outlined in the Design Document, the final product was a complete success. Below, in Figure 12, is a photo of the robotic arm just after successfully completing each of the tests to pick up a wooden block.

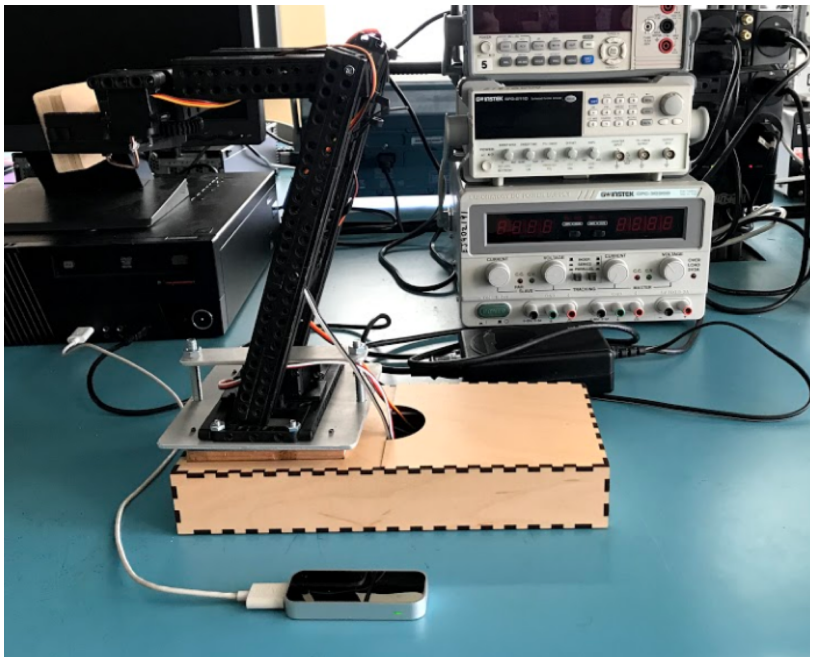

Figure 13 - Robotic Arm After Completion Of Testing

The LEAP Motion test was passed as every linear coordinate of the user's movements were properly recorded, calculated, and sent to the Arduino. The Arduino Microcontroller properly processed the data and controlled each servo accordingly and accurately.

Process Outcomes

Assumptions

At the start of the project, the team assumed a few things. Each assumption is listed below with a short summary detailing the issue and how it was fixed.

LEGOS:

Originally the team planned to build the entire robotic arm with LEGOs. This seemed like the easiest solution because the team would not have to worry about interfacing LEGO pieces with non LEGO materials. However, this quickly changed after a few design iterations. The original LEGO rotating base did not provide enough support to keep the robotic arm stable while under regular use. The LEGO base also caused issues with 360 degree servos, which are covered in the next section. After preliminary testing, the rotating base was redesigned to what can be seen in the final product. The wood and aluminum provide the necessary support as they are stronger materials than plastic.

360 Degree Servos:

Initially the team planned to use 360 degree servos to move both the rotating base and claw. This was due to the initial design being completely made out of LEGOs and requiring gearboxes. Due to the way LEGO gears are designed, more than one rotation would be needed to rotate the base and to open and close the claw. This led the team to purchase 360 degree servos for the base and claw. However, the team faced another design constraint when attempting to interface the servos with the Arduino. Unfortunately, it is impossible to rotate a 360 degree servo to a specific angle with the framework that the team had set up. Since this was critical to the rest of the project, the claw and base were redesigned to what can be found in the final product. The new design makes use of 270 and 180 degree servos which are able to be rotated to a specific angle.

Wrist Control:

The team also assumed the wrist would be controlled by the user moving their own wrist up and down. However, due to the way the LEAP tracks the user's arm, this was not possible. Motion is tracked by looking at two points on the hand and calculating the distance between them. For example, to close the claw the LEAP tracks the user's pointer finger and thumb. The distance between these fingers determines how much the claw closes and opens. After testing, the team found there were not two points for the user's wrist to track. Instead of bending the user's wrist to control the wrist servo, the team chose to use the z-plane. The user move's their arm forwards and backwards to move the wrist servo inward and outward. This is not a perfect solution, but it is the most user friendly solution based on what the LEAP Motion Controller is capable of.

LEAP Tracking:

In addition to issues with controlling the wrist, the team also ran into issues with the LEAP tracking. Initially the team assumed the LEAP would have perfect motion tracking. After a few days of testing, it was clear that this is not the case. There were two main issues with how the LEAP tracks hands. If the user gets too close to the LEAP, such as when attempting to lower the forearm, the LEAP will lose track of the user and cause the robotic arm to stop moving. Secondly, the team ran into issues with the LEAP's finger tracking ability. Occasionally when the user's fingers were too close to each other, the LEAP would lose track of the fingers and the claw would stop working. The forearm issue was fixed after altering the LEAP's range. By increasing the lower bound of the range where the LEAP begins tracking, the user does not have to lower their arm as much to move the forearm. The claw issue was fixed by altering how much the user needs to close their fingers in order to control the claw. Finally, the team added a sleep function to the project. This was not initially in the functional specifications, but it added to the overall user experience. When the LEAP stops tracking a hand or detects more than one hand, the robotic arm returns to a resting position. The robotic arm then stays in its resting position until the LEAP registers a single hand or until the power is turned off.

Milestones

Table 2 - Milestones

Milestones	Intended Completion Date	Date Completed
Functional Specifications Document (v 1.0)	October 12, 2018	October 5, 2018
Final Budget	October 12, 2018	October 5, 2018
Complete a working Robotic Arm Prototype	November 9, 2018	November 20, 2018
Program a Preliminary Arduino Test Script	November 19, 2018	October 30, 2018
Test and Debug the Arduino Script	November 24, 2018	November 2, 2018
Test and Debug Arduino Script with the Robotic Arm	November 24, 2018	November 27, 2018
Design Document (v 1.0)	November 30, 2018	November 27, 2018
Shiley Showcase	December 7, 2018	December 7, 2018
LED and LEAP Interface	February 7, 2019	January 24, 2019
Single Servo and LEAP Interface	February 7, 2019	January 24, 2019
Full Servo and LEAP Interface	February 15, 2019	January 25, 2019
Final Report (v 1.0)	March 15, 2019	March 15, 2019
Arm Aesthetic and Housing	March 21, 2019	March 1, 2019
Test and Debug Final Design	March 29, 2019	TBA
Founder's Day Presentation	April 9, 2019	TBA

Risks

Table 3 - Risk Severity And Likelihood

Risk	Severity	Probability
Schedule Delays	Medium	Medium
Damaged Components	Medium	Medium
Assembly Errors	Medium	Medium
Code Failure	High	Low

Schedule Delays:
This was a medium probability risk because the varying team member class schedules caused difficulty in scheduling weekly meetings. Also, in the spring semester, ordering the LEGO pieces had some unforeseeable shipping delays. However, since the team remained proactive throughout the course of the project, the team was actually able to stay ahead of schedule and finish the design part of the project much earlier than expected.

Damaged Components:
There was a medium probability that components could have been damaged when assembling the servos and components. Although powering the servos was not a problem, overloading the servos or putting too much stress on them during construction could have occurred since the team had minimal experience with servos. However, the team handled this by operating the servos within the specified current rating of each servo.

Assembly Errors:
Putting together the different components that each team member worked on involved a moderate amount of risk. To minimize this, each team member documented all design and assembly progress they made. This allowed for easy troubleshooting, and overall reduced the severity of any issues.

Code Failure:
The team was unlikely to fail in completing codes for both the Arduino Microcontroller and LEAP Motion Controller. Even though the team had no previous experience coding in Javascript, through continuous research, the team was able to successfully produce code to interface the LEAP Motion Controller with the Arduino Microcontroller. Had the code not worked in the end, the impact to the project would have been severe.

Conclusion

The IR Motion Tracking Robotic Arm project handles inputs from the user, processes the input commands, and moves the robotic arm accordingly. The control input from the user is from an IR tracking sensor and the arm movement is achieved through moving four servos, thus resulting in four degrees of freedom. Team IR Motion Tracking Robotic Arm completed all their milestones, and is ready for Founder's Day Presentations. In the end, the IR Motion Tracking Robotic Arm is capable of moving a wooden block that is 1.5 cubic inches in size and 0.5 ounces in weight. It is capable of picking up the wooden block, turning 180 degrees, and then placing down the block. As a result, this project was a success.

Appendix A — Javascript Code To Convert LEAP Data To Servo Angles

```
/*
 * @Author: Team IR Motion Tracking Robotic Arm
 *
 * This is the code to control the robotic arm through the LEAP Motion
 * Sensor to the Arduino microcontroller
 *
 */

// load math.js
const math = require('mathjs')

//define LEAP ranges
LEAP_ranges = Array(
                Array(-100,200), //x
                Array(90,650), //y
                Array(-300,300), //z
                Array(20,70)//
                );

//define servo ranges
servo_ranges = Array(
                 Array(10,169), //base
                 Array(100,170), //elbow
                 Array(50,179), //wrist
                 Array(0,179) //claw
                 );

// grab the LEAP data
var webSocket = require('ws'),
ws = new webSocket('ws://127.0.0.1:6437'),
five = require('johnny-five'),
board = new five.Board(),
// mm range of LEAP motion to use, see LEAP-range.js to find
frame, palm = Array();

// parse the data and respond
board.on('ready', function() {
        // setup four servos on the four pwm pins of the Arduino
        servos = Array(
                    new five.Servo({ pin: 12 }), //base
                    new five.Servo({ pin: 11 }), //elbow
                    new five.Servo({ pin: 10 }), //wrist
                    new five.Servo({ pin: 9 }) //claw
                    );

        //initialize servo
        servos[0].to(80, 800); //base
        servos[1].to(130, 800); //elbow
        servos[2].to(101, 800); //wrist
        servos[3].to(101, 800); //claw

        ws.on('message', function(data, flags) {
            frame = JSON.parse(data);
            //check if only one hand is in the LEAP
            if (frame.hands && frame.hands.length == 1) {
            // extract center palm position in mm [x,y,z]
            palm = frame.hands[0].palmPosition;

            // map x,y,z positions of LEAP to servos
            for (axis=0; axis<3; axis++) {
            servos[axis].to(palm[axis].map(axis), 150);
            }

            //calculate finger distance
            if(frame.pointables.length > 1) {
            f1 = frame.pointables[0];
            f2 = frame.pointables[1];
            fingerDistance = distance(f1.tipPosition[0],f1.tipPosition[1],f1.tipPosition[2],
                    f2.tipPosition[0],f2.tipPosition[1],f2.tipPosition[2]);
            clawAngle = servo_ranges[3][1] - fingerDistance +25;//add 25 to make the claw close more
            //check that the angle is within range
            clawAngle = (clawAngle > servo_ranges[axis][1]) ? servo_ranges[axis][1] : clawAngle;
            clawAngle = (clawAngle < servo_ranges[axis][0]) ? servo_ranges[axis][0] : clawAngle;
            servos[3].to(clawAngle, 150);
```

```
            }
            }
            else{
            //initialize the robotic arm if no hands are present
            servos[0].to(80, 800); //base
            servos[1].to(130, 800); //elbow
            servos[2].to(101, 800); //wrist
            servos[3].to(101, 800); //claw
            }
            });
        });

// map two number ranges
Number.prototype.map = function (axis) {
    var output = math.round((this - LEAP_ranges[axis][0]) * (servo_ranges[axis][1] - servo_ranges[axis][0]) / (LEAP_ranges[axis][1] - LEAP_ranges[axis][0])
+ servo_ranges[axis][0]);

    // check output is within range, or cap
    output = (output > servo_ranges[axis][1]) ? servo_ranges[axis][1] : output;
    output = (output < servo_ranges[axis][0]) ? servo_ranges[axis][0] : output;
    if (axis==0){
        output = servo_ranges[axis][1] - output;
    }
    if (axis==2){//wrist
        output = output * 1.05;
        //console.log("Wrist Angle: ", output);
    }
    return output;
}


//function to calculate distance between two fingers
function distance(x1,y1,z1,x2,y2,z2) {
    return math.sqrt(square(x2-x1)+square(y2-y1)+square(z2-z1));
}

//function to calcualte the square of a number
function square(x) {
    return x*x;
}
```