**The Animatronic Hand Project**

**Dingfeng Shao***,* [*dishao@syr.edu*](mailto:dishao@syr.edu)

**Junfeng Que***,* [*jque@syr.edu*](mailto:jque@syr.edu)

**Ho Yin Wong***,* [*hdwong@syr.edu*](mailto:hdwong@syr.edu)

**Zilong Li***,* [*zli18@syr.edu*](mailto:zli18@syr.edu)

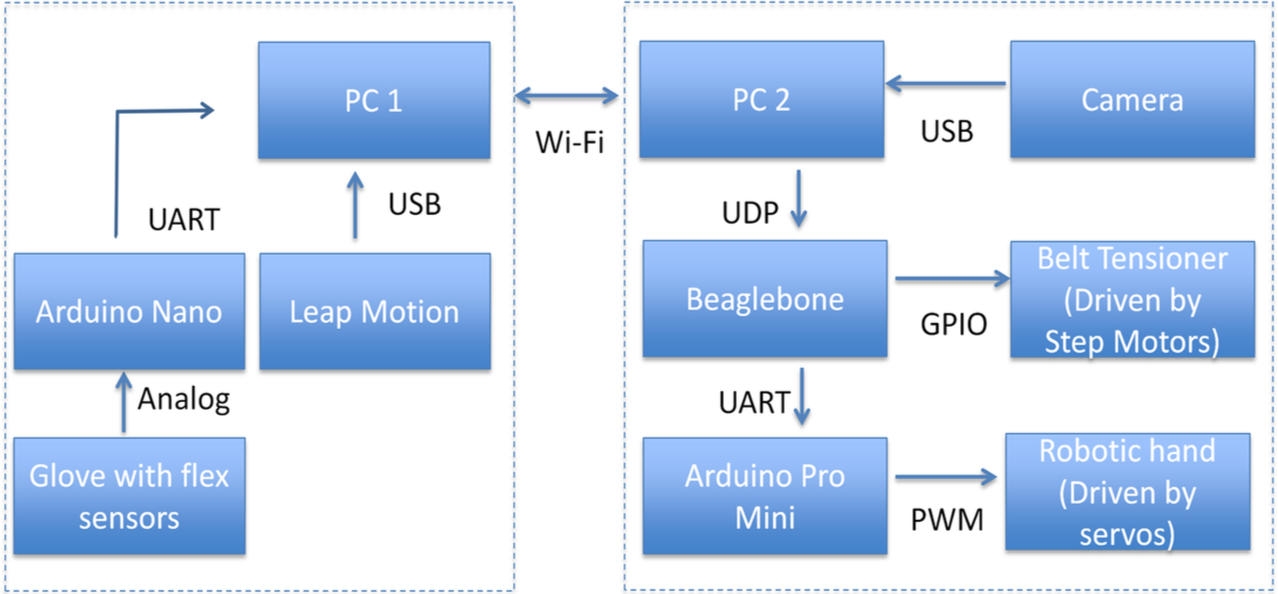
**College of Engineering & Computer Science, Syracuse University**

**Abstract**

This report describes how the Animatronic Robotic Hand Project was designed and built based on the concept of animatronics. The robotic hand was able to move freely within a three-dimensional space based on a belt tensioner system and a high torque servo, and it could be controllable by a customized glove with flex sensors and two IR cameras from Leap Motion through Wi-Fi communication, and with the combination of wireless video stream feedback, this project allows people to perform and monitor dangerous experiments from a safety distance after further development. In addition, the robotic hand could be able to imitate and perform the user’s hand gestures through a web camera by applying OpenCV. Besides the two applications above, we also added another function called Auto-painting, which allowed the robotic hand to draw alphabetic letters and small images pixel-by-pixel automatically. All the software implementations and hardware components of the Animatronic Hand Project were explained in this report with great details.

**Introduction**

Animatronic Hand is a robotic hand with flexible joints, so it could be the substitution of human hand to operate the experiments. The robotic hand is wirelessly (Wi-Fi) controlled by a customized glove, which is connected with five flex sensors. And these sensors are used to monitor the movement of human fingers. Therefore, after putting on the glove, the user’s fingers movement can be immediately performed on the robotic hand. We also used Leap Motion, which includes two IR cameras, to capture the user’s hand movement in order to control the movement of robotic hand. By using this project, scientists who have to work on dangerous experiments could be able to operate those experiments from a safety distance.



**Method**

* Beaglebone Black and Arduino

On the robotic hand side, we chose Beaglebone Black as our microcontroller. We had to control 6 servos to drive 5 fingers and the arm, but Beaglebone did not have enough PWM ports. Thus we added an Arduino Pro mini as an extension. Beaglebone sent the UART signal to the Arduino Pro mini first and then Arduino decoded the signal, mapped data, and drove servos to different angle using PWM. Meanwhile, Beaglebone directly controlled three stepper motors through GPIO. In Beaglebone black, we wrote a python program. First, it received data from PC. Since it would take time for stepper motors to rotate, we implemented multithreading to separate servos from stepper motors. Because of Beaglebone was less powerful than PC, we had to slow down the transmission. Thus, we used TCP to ensure performance. Also, after the PC sent a signal, we made it wait for the confirmation. Beaglebone would send the confirmation after it finished moving the stepper motors, sent out the UART signal to Arduino.

* Glove and Arduino

On the glove side, we used Arduino Nano to gather data from flex sensors. We sewed one flex sensor on each finger and put a small breadboard on the back of the glove. We built voltage divider on the board and connected it to the flex sensors and Arduino. Arduino read data from sensors, decoded the data and then sent to PC using UART. Since the flex sensors were fragile, we used heat shrink tube to protect them.

* Leap motion

The Leap motion system is used to recognize and track the position of hand. The device operated in an intimate proximity with high precision and tracking frame rate, and it could report discrete position. There are two sections for the Leap motion application: frame listener and data analysis. For frame listener section, we created controller object to connect to the device and used frame object to obtain all the tracking data in the Leap motion system. Those tracking data included frame id, frame timestamp, hands property and gestures. For the other section, the application analyzed the hand data for control command. The Leap motion control commands were used to control the horizontal, vertical, forward and backward movement of the robotic hand. In order to get the accurate vertical wrist data, the program calculated the hand direction for pitch angle in degrees. For the other control command, the program used the leap motion to map the 3D coordinate systems as a virtual space. The program mapped the coordinates to 3D system with the origin of the interaction area at the bottom center of the world, scales the range of the interaction box to 200 3d units. The Z coordinate is the backward forward measurement, and he X coordinate is the horizontal measurement. In complement to this Leap motion control command, we also built a GUI for the user to observe.

* Robotic Hand

Each part of the robotic hand was 3D-printed (Gael, 2015), and we used cable wires to connect each joint to another. In order to control the movement of the robotic hand fingers, we put two fish lines through each finger, from fingertip to wrist, pulling one of the fish line could straighten the finger, and pulling the other fish line could curl the finger. We used one servo to control one of the five fingers of robotic hand, but before we connected the fish lines to the servo, we 3D-printed a cover for the servo wheel. The cover had a groove around its side, so the fish line could go around the wheel in the groove when we spin the servo. We set the wheel of a servo to be 0 degree, and bound the both fish lines on the both end of the groove when the fish lines are tense and the figure is straight. Therefore, when the servo started spinning toward 180 degrees, the fish line used to straighten finger could lose its tension, and the other fish line could wind into the groove and pull the finger to curve. And when the servo was spinning toward 0 degree, the fish line used to straighten the finger could gain its tension and pull the finger to be straight.

* Belt Tensioner System

In order to make robotic hand be able to grab and move certain objects around, we built a belt tensioner, and this system could uphold and carry the robotic hand to move toward X, Y and Z directions. For the X-axis movement (Left & Right), we 3D-printed and assembled the X-axis of Prusa i3 3D-printer (RepRap, 2014). For the Y-axis movement (forward & backward), we 3D-printed another two set of X-axis, then we fixed one of the carriage from these two set of X-axis with the X End Motor, which was previously printed for the X-axis movement, and we fixed the other carriage with the X End Idler. Therefore, we could move the X-axis forward and backward by sending the same data to the two step motors, which were used to drive the other two set of X-axes. For the Z-axis movement (Up & Down), we set a high torque servo horizontally using a set of pan tilt on the X-axis carriage, so the robotic hand could be lifted and lowered by the horizontal high torque servo while moving towards X and Y axes.

Finally we used a 20cm pipeline to connect the horizontal servo and the robotic hand as the forearm, and we put the five servos, which were used to control the fingers movements of robotic hand, on the top of the horizontal servo. We also use a large spring to connect the carriage, which was used for X-axis movement with the horizontal servo, so the large servo could be stabilized on the carriage without worry about the weight of robotic hand and the five servos.

* Server PC

This PC was used for the client PC to communicate with the robotic hand. We wrote a C# program with multithreading. The receiving thread received the data from the client PC, which included 6 integers for servos and 3 integers for stepper motors. Then 3 sending threads would send servo data, x-stepper motor data, and y-stepper motor data to Beaglebone through TCP. After it sent signals, it would wait for the confirmation from Beaglebone. Otherwise, the buffer would be fulfilled and Beaglebone could not get the real time data. Also, C# program had a simple GUI to show the data it received. Besides, we implemented a special function for the hand: auto-painting. We used MATLAB to filter English alphabets images and converted them to 10\*10 Boolean matrixes (1 is black and 0 is white). In the GUI, when the user pressed the Auto button, the server PC would not use the data from the client PC. After the user entered the letter they wanted the hand to paint and pressed Start button, C# program would read the corresponding file in the computer. Then it would send pre-designed data to Beaglebone to control servos and stepper motors to draw the image pixel by pixel.

* Live Video Streaming

For the live video streaming, we used the bitmap source conversion to display the actual frame from the server. In order to maintain the high performance and the fluent transmission, the program kept the each frame to 480 x 320. Then the application converted the regular image to a jpeg data format and did a buffer size check. Then the jpg data converted to the byte format and send to client program by UDP socket. To keep synchronization, both send and receive end used a timer that active per millisecond. The receive program received the data from the memory stream and converted the stream to a displayable image in bitmap source format.

* Hand Gesture Tracking

The hand tracking was based on the color recognition using OpenCV. The program was initialized by sampling color from the hand. The hand was then extracted from the background by using a threshold using the sampled color profile. In this program, we also created the HSV (Hue Saturation value) base skin color detector to filter out the background and hand. Then the application made binary representation of hand and get the largest convex points in contour in the camera frame. The confirmed convex points were calculated to get convexity defects and filter out convexity defects that are not relevant. In order to get a better sense of the convex points around the hand, the application painted the tips, finger depth, and hand sharp. The other part of this program was used to determine the hand gesture with the given convex points. In the algorithm perspective, the application set the fixed point for the five fingers in the initial position. The thumb, index, middle, ring and the little finger were count from left to right in sequence. Then the application sampled 20 frames per detection for each gesture. Once the fingers bended, the corresponding finger would be set to zero.

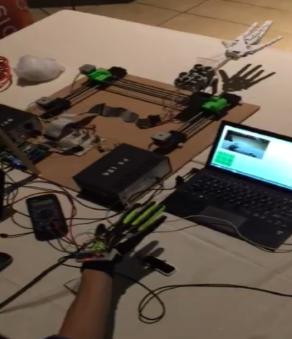
**Results**

* Communication

First of all, the communication worked well. The signals were transmitted without packet lost from server PC to Beaglebone, and from client PC to server PC. Also, thanks to multithreading, servos and stepper motors functioned well, which means that they did not affect each other. However, there were still some delays in stepper motors, because we could not make stepper motors fast and powerful at the same time. In order to move the hand precisely, we had to lose some speed and make stepper motors slower to gain more torque.

* User Control

The client program could successfully send data to the server program through Wi-Fi, and the server application was able to recognize the received data and duplicate the same hand movement and hand gestures from the user. The user was required to wear a glove to detect his or her finger bending. The user also had to put his or her hand above the Leap motion sensor with at least 15cm for the best performance of the hand position measurement.



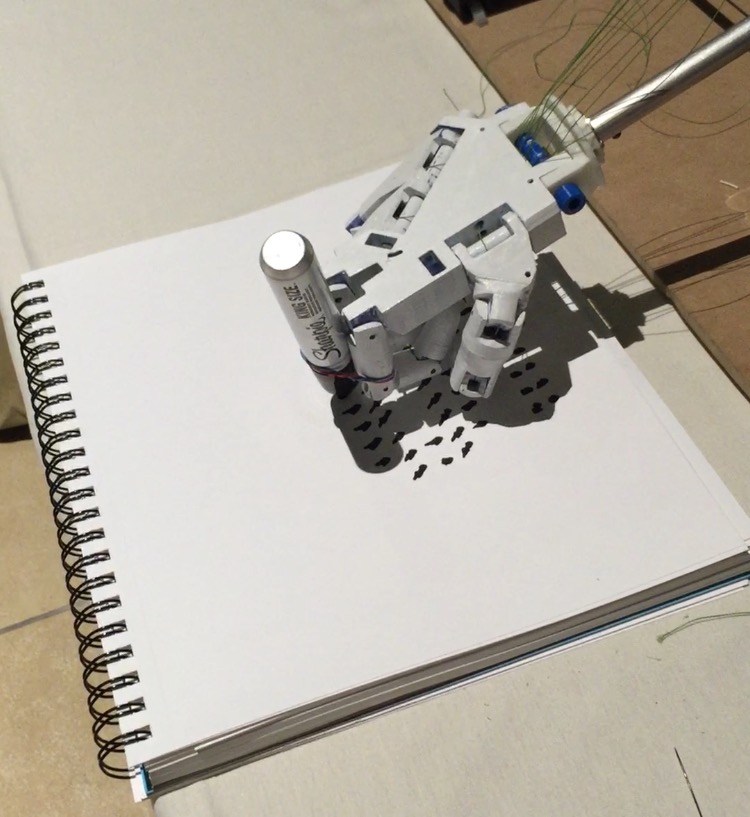
* Imitation

The Imitation function was successfully implemented. When the user put his or her hand in front of the camera. The application was able to subtract the background based on the human hand skin color. As the imitation began, the user’s hand needed to stay at the fixed position in order to obtain the initial fingertips position for the five fingers. This application was required to calibrate the HSV filter by the user due to the background light and background color manually. For the result, robotic hand was able to do the same gesture as the user hand with 0.5 sec delay due to the sample correction.



* Auto-painting

The auto-painting function was successfully implemented. When the user entered the letter and pressed the Start button, the hand could automatically draw the recognizable letter on a paper clearly. However, since the stepper motor moved quite slow, it might take 40s to finish one letter. Also, when the arm moved up and down, the pen it held was not quite stable. Thus, the dots on the paper were not quite neat.



**Discussion**

The Animatronic Hand Project was successful and we were impressed by the performance of all three applications, which were User Control, Imitation and Auto-painting. For the software implementation, we used C, C# and Python to describe and accomplish multiple functionalities, which includes GPIO control, UART communication, TCP, UDP, multi-threading, GUI design, OpenCV video transmission and data mapping. Although we had experience among all these fields, we still spent a lot of time to modify the code in order to make everything work perfectly together without communication and data synchronization errors.

For the hardware components, except the engineering ideas and skills from our own, we also learned from other mechanical structures, such as Prusa i3 (RepRap, 2014), and applied them to our design. However, we did not applied the Y-axis design from Prusa i3, because we wanted the Y-axis to move at the same speed as X-axis, and the screw threading design form Prusa i3 had a slow movement speed. In addition, we also spent quite some time to adjust the design in order to make the robotic hand flexible but also have enough strength to pick up certain objects, and the tensioner system could be stable during the operation. Therefore, after using screws and nuts to combine everything together, we also use several strong zip ties to locate the whole belt tensioner system on a hard wooden board to increase its stability.

However, there were still two aspects that could be improved, and the first one was the wrist rotation. Since the robotic hand and the five servos combination were already fairly heavy, adding another servo to rotated the robotic hand, and its relevant 3D-printed component, which was used to carry the servo and connect it with the robotic hand, might force us to replace the current horizontal servo to another one for higher torque, so we had to give up the rotational robotic wrist due to the insufficient time. The other improvable component was the customized glove. We used the IR cameras inside Leap Motion to capture the user’s hand movement, and this design had the best performance while the user was using bare hand. Although we used the glove with the thinnest clothe that we could find on the market, Leap Motion still sometimes lost track of the glove. This problem might be solved in the future by using special material on the glove or replacing Leap Motion with other better IR cameras.

**Reference**

Gael, L. (2014). Finger Starter. *InMoov.* Retrieved from: <http://www.inmoov.fr/finger-starter/>

RepRap, (2014, November 26). Prusa i3 Rework X Axis Assembly. *RepRap.* Retrieved from: <http://reprap.org/wiki/Prusa_i3_Rework_X_axis_assembly>