Project Title: Classroom Robotic Development Platform

Team Members: Corey Hooker, CmpE & EE
Christopher Minning, EE
Daniel Moening, CmpE

Faculty Advisor: Carlos Pomalaza-Ráez, Ph.D

Date: April 22, 2016
Acknowledgements

We would like to express our deepest gratitude to our primary advisor, Dr. Carlos Pomalaza-Ráez. Thank you for being available at all hours of the day, and for all the Saturday’s you spent, guiding us through a successful project.

We would also like to thank our Co-Advisor, Dr. Yanfei Liu, who provided expert knowledge of Robotics and thoughtfully answered all of our questions and provided advice when we needed it. Thank you.

A special thank you goes to Mr. Bill Westrick and Dr. John Licato, for all of your time, knowledge and patience throughout this project.

Finally, we would like to thank the IPFW Engineering Department for sponsoring this project.
# Table of Contents

Acknowledgements ......................................................................................................................... 2  
Abstract ........................................................................................................................................... 4  
Section I: Problem Statement .......................................................................................................... 5  

  1.1 Requirements and Specifications ..................................................................................... 6  
  1.2 Design Variables .............................................................................................................. 6  
    1.2.1 Hardware ................................................................................................................... 6  
    1.2.2 Software ................................................................................................................... 7  
  1.3 Limitations and Constraints ............................................................................................. 7  
  1.4 Additional Considerations ............................................................................................... 7  
  1.5 Safety ............................................................................................................................... 7  

Section II: Detailed Design of Chosen Concept ............................................................................. 8  

  2.1 Functional Overview of Robotic Arm ............................................................................... 9  
  2.2 Electrical Schematic ..................................................................................................... 11  
  2.3 Octal Buffer Pinout ...................................................................................................... 12  
  2.4 Serial Timing ............................................................................................................... 13  
  2.5 Mechanical Robot System Layout ............................................................................... 13  
  2.6 Power Supply System .................................................................................................. 14  

Section III: Final Built Design ...................................................................................................... 16  

  3.1 Final Electrical Design ................................................................................................... 17  
  3.2 Power supply ................................................................................................................ 17  
  3.3 The Robot ...................................................................................................................... 18  
  3.3 Octo-Buffer Chip ........................................................................................................... 20  
  3.4 The Arduino Mega ....................................................................................................... 22  
  3.5 Raspberry Pi Message Server ........................................................................................ 24  
  3.6 Raspberry Pi Streaming Video Data .............................................................................. 26  
  3.7 Robot Control GUI for Windows ................................................................................... 26  

Section IV: Cost Analysis & Testing ............................................................................................ 28  

  4.1 Cost Analysis ................................................................................................................ 29  
  4.2 Testing ............................................................................................................................ 29  

References ..................................................................................................................................... 32
Abstract

The focus of this project is to develop a robotics platform for the teaching and research of robotics and artificial intelligence.

As such, a robot shall be developed that can search and pick up an object of up to 300 g in weight, that can fit in a five inches’ cube, and place it in a different location. The robot shall complete this task on its own, given a wireless command. The robot shall incorporate sensors, such as visual and pressure sensors to accomplish this task.
Section I: Problem Statement
1.1 Requirements and Specifications

1.1.1 Programmability - The robotic system must be able to be programmed using a high level programming language to complete an action via Remote HMI.

1.1.2 Sensors - The robotic system must be equipped with visual sensors for object detection, sensors for grip feedback, and motor feedback.

1.1.3 Wireless Standard - The robotic system must be able to receive and transmit commands over a standard IEEE wireless protocol from the robot controller to the remote HMI and vice-versa.

1.1.4 Purpose - The robotic system will be used in a class setting to demonstrate the basic electromechanical controls of a robot.

1.1.5 Budget - The total cost of the project must be less than or equal to $1000.

1.1.6 Pick-up Weight - The robotic system must be able to lift 300 grams and manipulate the objects placement.

1.1.7 Object Location Displacement - The robotic system must be able to pick up and move the target object from one location to another.

1.1.8 Gripper Size – The gripper will need to have an internal gripping width of at least 5 inches.

1.2 Design Variables

1.2.1 Hardware

1.2.1.1 Construction Materials - Materials can be ordered from a supplier or manufacturer. Manufactured components could be created utilizing a 3D printer, CNC, or manual material fabrication tools (i.e. mill, lathe, and etc.).

1.2.1.2 Power - The robot will require an electrical source such as that provided by an electrical outlet.

1.2.1.3 Robot Controller - Computing module physically connected to the robot motors and sensors in order to capture sensor, control motor actions, and communicate with the Remote HMI.

1.2.1.4 Gripper - The robot should have the ability to secure the object, without damaging it, when moving the object from one location to another.

1.2.1.5 Remote HMI - A computing module that wirelessly communicates with the Robot Controller. This allows the user to send and receive control information to and from the Robot Controller.

1.2.1.6 Data Storage - Robot must be able to store data containing control information, sensory information, and image data.

1.2.1.7 Heat Dissipation - The unit’s robot controller must remain in its operational heat range so it performs consistently without damage.

1.2.1.8 Receiver/transmitter - A communication module utilizing an IEEE wireless standard will need to be on the robot controller and remote HMI to receive and transmit data.
1.2.1.9 Motors - motors will be used to move the various parts of the arm based on their respective torque requirements for weight pick up.
1.2.1.10 Electrical Sensors - Provide sufficient electrical feedback to the robot controller so as to control the robot effectively.

1.2.2 Software
1.2.2.1 Robot Controller Operating System - A sufficient platform for programs to be developed and implemented so as to control and monitor robot movements.
1.2.2.2 Wireless Communication - Software will be fitted with an IEEE wireless standard in order to communicate with the remote HMI.
1.2.2.3 Programming interface - The programming interface should allow for easy to use and should allow playback of stored programs and video feedback.

1.3 Limitations and Constraints
1.3.1 Durability - This robot will be used in the classroom with large attendance of people and may be moved between multiple classrooms or areas. The robot must be able to withstand the strain of being moved from place to place. The material must also sustain the weight of the object.
1.3.2 Weight - The robot must be less than or equal 30 lbs. making it light enough to be moved through a multi-story building.
1.3.3 Time - The design, building, and testing of the robot must be completed within two semesters.
1.3.4 Power Consumption - Power will be supplied by standard outlet found in the classroom, and conditioned from there, to each component requiring power. The power consumed by the robot’s components must be within the limit of this source.
1.3.5 Data Acquisition Resolution - The resolution of the input and output of the robot controller needs to be of proper granularity to ensure precision.
1.3.6 Motor Strength: - Motors will need to be capable of moving the mechanical components of the robot as well as lift the 300-gram object.
1.3.7 Conductor Size - Conductors should be large enough to safely carry the current.

1.4 Additional Considerations
1.4.1 Display - A display panel (i.e. LCD, LED, etc.) may show relevant information to assist debugging by the user.
1.4.1 Buttons - Buttons on the robot controller may assist the user in display manipulation.

1.5 Safety
1.5.1 Electrical Safety - There should not be exposed uninsulated conductors which could cause harm to the user.
1.5.2 Emergency Stop - A device to capture user input should be hard wired, in such a way that it can stop the robot in the event of an emergency.
1.5.3 Robot Safety Communication - The robot should be able to communicate back to the user, whether or not a given command can or cannot be executed, given a predefined set of “rules” that prohibit harmful actions to the robot.
Section II: Detailed Design of Chosen Concept
2.1 Functional Overview of Robotic Arm

With our chosen concept, a brief overview of how the robot will operate can be seen below, in Figure 1. The camera, connected to the end effector of the robotic arm, will communicate with the Robot PC to transmit the images it captures, and receive commands such as image capturing or image resolution alterations. For autonomous operation, the Robot PC will take the results from the Camera, and communicate necessary commands to the Micro Controller. The Microcontroller communicates with the Pressure sensors in the robot and the servos, to ensure that the object is either obtained by the end effector, or not. As this occurs, the Microcontroller is constantly updated with feedback from both the sensors and robot, and communicates any pertinent information to the Robot PC. Once the object is deemed to be obtained, the command for movement and placement of the object, given by the PC, is carried out. Communication between the Robot PC, Camera, Sensors, Microcontroller, and Robot, occur until the command is deemed satisfied.

Figure 1. This is a basic overview of connectivity of the components that will be making up the robot or communicating with the robot.
Figure 2. Overview of detailed robot operations and checks. This covers the interaction of the program and robot, through natural operations, and the error reporting.
2.2 Electrical Schematic

In our electrical design, we show the seven servo motors that are integrated into the frame of the Robot. These motors are “daisy-chained” and powered by a Dual Voltage Power Supply that outputs 12V at up to 10.5A, and 5V at up to 2.5A. The data from the motors are fed through a 74LS241N Octal Buffer Chip that allows for ease in communication back and forth between the Arduino Due and the motors. The Arduino Due and Raspberry Pi are connected together with the Raspberry Pi connected to 5V and the Arduino Due to 12V. The Camera that is mounted on the Robot is powered by the Raspberry Pi. We incorporate a FlexiForce Pressure Sensor that feeds pressure data into the analog input of the Arduino Due. This is powered by the 5V output.
on the Arduino Due. The user will control the robot using a computer that wirelessly communicates with the Raspberry Pi.

2.3 Octal Buffer Pinout

As shown above in Figure 3 the servo data line will connect to the 74LS241N octal buffer chip. The chip does many things that would take more components and time to implement connecting directly to the Arduino Due. The chip acts as a buffer to the Arduino for any errors in communications. The chip also implements the pull up resistor for the communication of the servos, where the line is only pulled low for Zeros and left high for Ones. The chip also allows us to control the flow of data. The enable pins, 2 and 19, will be connected and when high the data will flow from the Arduino Due to the servos and when it is low the data will flow from the servos to the Due. The A pins flow to the corresponding Y pins. This can be seen in the IC symbols shown in Figure 4; furthermore, the pins are shown in Figure 5. The top arrow in Figure 5 illustrates reading from the servos and the bottom arrow the sending to the servos.

![Octal Buffer Circuit Diagram](image)

**Figure 4.** The octal buffer circuit demonstrating the direction control using pin 1G and 2G. Since the data to and from the servos is asynchronous each servo daisy chain network will be connected to both pins of Y1, Y2, Y3, and Y4. The Arduino’s serial pins will be connected to A1, A2, A3, and A4 (Texas Instruments, 2010).
Figure 5. The top arrow shows that when PIN2 is low data will flow from 1A1 to 1Y1. The bottom arrow illustrates that when PIN19 is high that data will flow from 2A3 to 2Y3.

2.4 Serial Timing
The baud rate for the communication ranges from 9615.4 bps to 1Mbps depending on the amount of data to be sent. A communication speed value from 0 to 254 can be used to set the baud rate and the speed can be calculated using the formula below (Robotis, 2016).

\[
\text{Speed (bps)} = \frac{2000000}{\text{speed value} + 1}
\]  

A delay between the servo receiving instructions and sending a status may also be set. The range of the delay value is from 0 to 254. Each increment is a delay of 2 microseconds. Therefore, a delay of 10 microseconds would have a delay value of 10 or 0xA in hexadecimal. This will be important in order to allow the microcontroller enough time to switch between transmitting and receiving modes.

2.5 Mechanical Robot System Layout
In the figure below the completed system layout is demonstrated. The Arduino Due and the Raspberry Pi 3 are stacked on top of each other in order to save space. The Raspberry Pi also has the Wi-Fi receiver on board and it would be best to keep that as high as possible to prevent radio interference from other devices. A plate is needed between the Arduino Due and the Raspberry Pi because their mounting holes do not align with each other. An acrylic plate may be used due to its low cost and availability. The power supply is the furthest from the robot base in order to easily route wires to power the Raspberry Pi and the AX-12A servos on the robot. Wires to the robot may be ran underneath the Raspberry Pi and Arduino Due since they are on standoffs which provide plenty of clearance underneath. The standoffs are 3mm female and 3mm male, allowing them to easily connect with each other. They are 25mm long which will be
long enough for the wires to connect to the Arduino Due underneath the Raspberry Pi. Due to the 300 degree turning radius of the robot, there are no concerns of the robot colliding with the computing hardware along the horizontal axis. However, if the robot were to reach straight back over itself, a collision could occur. Software limits will need to be placed in order to prevent this.

![Diagram of robot and components layout](image)

Figure 6. This is a detailed design of how the robot and other components will be laid out. On the front of the plate, upper left-hand side, is the robot base. And moving back is the Pi3 mounted on top of the Arduino and then the power supply.

### 2.6 Power Supply System

![Power supply diagram](image)

Figure 7. This is an overview of the designed power supply. With LT Spice, we use a transformer with a split secondary, to produce the desired outputs for our Robot and our Raspberry Pi.
The power supply design concept uses a center tapped transformer to "split" the voltage of the secondary into two separate voltages. At the secondary, the AC voltage is fed through a Full Bridge Rectifier circuit, and then smoothed in both its voltage and current, to supply a DC voltage to specific load requirements. The Robot requires 10.5V with up to 10A in current, while the Raspberry Pi requires 5.5V at up to 2.5A. Using an inductor in series with the load, on each section of the circuit, we are able to provide a fairly linear output voltage and current for each load. It is desired to block as much of the AC aspect of the current as possible from reaching the load. An inductor opposes changes in current and readily allows DC current to the load, which then allows for a smooth DC Voltage across the load. Efficiency of this Power supply design is approximately 81.2%. Improvements will be made over the summer to add voltage regulation to the power supply, to maintain the voltage no matter the load. Turns ratios modeled in LT Spice are based on a square proportion to ideal turns-ratios. The turns-ratios used are approximately 2.5:1 for the R_Robot Load and approximately 5.75:1 for the R_RaspberryPi Load.

\[
\frac{N_1}{N_2} \approx \sqrt{\frac{L_1}{L_2}} \tag{2}
\]

\[
\eta = 100\% * \frac{P_{in}}{P_{out}} \approx \frac{8}{\pi^2} * 100\% \approx 81.2\% \tag{3}
\]
Section III: Final Built Design
3.1 Final Electrical Design

As can be seen in Figure 9, the Arduino Due has been replaced with an Arduino Mega. This is due to issues with code implementation on the Due. The motor connection scheme is different in that all seven motors are now “daisy-chained” for ease of cable management, and only one transmission/reception pair for the 74LS241N chip is used to communicate between the Arduino and the motors. This connection scheme was found to have no visible effect on speed with communication to and from the motors. Rather than use the Waveshare Fisheye camera as chosen in our prototype, the Raspberry Pi Camera is chosen for its “plug and play” adaptability with the Raspberry Pi board. The Arduino Mega is powered by our external power supply, and the 74LS241N chip is powered by the Arduino Mega.

![Figure 9: This is the final electrical diagram for the Robot.](image)

3.2 Power supply

Time and future classroom use was a deciding factor in choosing to go with a purchased power supply. The RD-125 model was chosen for its cost and criteria satisfaction. The power supply has two outputs, one 5V, the other a 12V. The 5V connection can supply up to 15A of current, while the 12V can supply up to 10A. The power supply has a safety feature that won’t allow the
power supply to work without having a minimum current draw out of each output when plugged in to the outlet. The power supply uses a standard 120V three prong wall jack to be powered. A minimum of .5A is required at the 12V output and a minimum 2A output is required at the 5V output. When turned on, the idle resistance of the motors on the robot generate enough current draw out of the 12V; however, it is required to use a power resistor in parallel with the 5 V supply to allow the power supply to run even when the Raspberry PI was turned off. A 2.2ohm 25W resistor was chosen so that the minimum current draw from the 5V power supply is always provided.

![Figure 10: Above is the chosen power supply for the project](image)

3.3 The Robot

The motors are numbered one through seven as seen in Figure 11. One is the motor at the base and turns the entire robot left and right. Motors two and three are located on the shoulder of the robot and mainly control how far the gripper is from the base. Motors four and five are on the elbow of the robot, and motor 6 is wrist of the robot. These three motors mainly function to orientate the gripper into a position to align the gripper with an object to grasp. Motor 7 turns two gears that control the open and closing motion of the robot.
The robot arm configuration means this robot has five axes of motion which can be seen in Figure 12. The first axis of motion is the turning of the robot left and right. Motor one is the only motor that controls this movement. Axis two is the shoulder joint and includes motors two and three. Axis two is mainly controlling the length the arm can reach from the base. Axis three includes motors four and five in what is referred to as the elbow joint. Axis four includes motor 6 at the wrist joint. Axis three and four mainly control the position and orientation of the gripper. Axis five contains motor 7 and is the opening and closing of the motor. The axes can be seen in figure 2.

These directions hold true for motors one, two, four, six, and seven; however, since motor two and three and four and five face each other, they move in opposite directions. When motor 2 moves to 400, motor three moves to 623.

This robot configuration has not changed from the design phase. The motors on the robot are orientated as shown with the exception of motors four and five. The motors are flipped so what is
shown as the outside of the motor in this picture is on the inside and vice versa. This is why the axis 2 and 3 increases in different directions in relation to one another.

Figure 12: Depicts the axis of motions for the robot. The green or bottom set shows the base motor as axis 1. If the motor’s position is increased, it will turn right and vice versa. The other motors are similar. Axis two contains motors two and three, the shoulder. Axis three contains motors four and five, the elbow. Axis 4, top-center and yellow, is the twist of the gripper. And motor seven is axis five which has gears moving the gripper open and back or forward and closed.

3.3 Octo-Buffer Chip

As seen in Figure 13, on the left is two 3-pin molex connectors. There are four traces that connect to the chip in the middle. One line is for an individual motor, the other three are each tied to two parallel connected motors. The chip is powered by our 5V source and grounded accordingly. The board contains two 5-pin Molex connecters that allow the connection of an Arduino Due to the chip.
Figure 13: This is a schematic of the developed OctoBuffer Chip used to aid in the communication between the individual Robotic servos, and the Arduino Mega.

Figure 14 is generated using the website Upverter. A trace width of 20mils is chosen so that currents up to 5.95A can be safely used. The board material used to physically implement this design is FR4 with a substrate thickness of .5oz/ft². A ground pour of copper was chosen to aid in longevity of use and provide extra protection from electrostatic discharge. Pads are chosen to be 1.5mm in diameter for feasibility in drilling and soldering. The orange is the general outline of the parts that will be soldered.
Figure 14: This is the Board Layout of the OctoBuffer Chip.

3.4 The Arduino Mega

The Arduino Mega (hereafter referred to as “the mega”) is the external controller of the motors. The mega receives a character array from the Raspberry Pi. The character array is sliced into sections so that the mega can process the command. The mega first checks the third character for conditions such as torque, voltage, position, temperature, etc. The condition then can be broken down by the first character from the Pi which determines the kind of function we want to perform with the motors: (1) Move, (2) Set, and (3) Read.
Figure 15. The microcontroller that is implemented into the system is the Arduino Mega.

There are some that may only have one or two of the functions because of the nature of the condition. The voltage can be set or read but cannot be moved. Examples of setting the voltage range of motor one to be nine to twelve volts would be: $SIV, 9, 12;$. So the mega knows that it needs to set voltage of motor 1 by the first three characters. The mega then parses the strings for commas to find the parameters to fill out the rest of the function inputs.

The rest of conditions are stored in a common Pi_And_Arduino.h header file. The header file stores the delimiter bit ‘,’ stop bit ‘;’, baud rate, and other the other conditions. The header file defines these all as constants for inside of the code, but if communication in the Arduino IDE, one would need to know the corresponding character codes. The Pi_And_Arduino.h can be found in the appendix for a full list of the conditions.

The mega communicates to all of these motors through the TX1 and RX1 because the TX0 and RX0 are also connected to the USB port that is powering it and communicating to the PI. The mega had some issues powering its five volt line for the Octo-Buffer chip when there is not enough power being supplied to it by the PI. This issue was resolved when the variable V1 of the power supply was increased to 4.5V from 3.8V. The Octo-Buffer chip allows the separation of half-duplex asynchronous line that the motors use to communicate.
Most of the functions come from Savage Electronic’s Header file; however, some were created so that we could either write values to unavailable areas from the provided functions or read from functions that we had written to previously. A function to send or receive from the motors starts with sending two start bytes, an ID byte, a length of the bytes sent, a read/write/reg_write byte, a starting register, and all of the parameters to send if writing, and a checksum byte.

3.5 Raspberry Pi Message Server

The message server relays messages sent through the Ethernet sockets to the Arduino serial port. This works by utilizing multiple processing threads to allow the message server work as quickly as possible. Currently, the buffer size for both the serial and the Ethernet socket communication is 256 bytes. Based on our command structure, this is more than enough data to send to and from the robot. The largest amount of data transferred over TCP/IP is status updates from the robot since all servos are read simultaneously.

In one thread, the server listens on the desired socket for incoming messages from a computer. When listening on the socket the operation is blocked meaning the program does not continue until a message is received. This made the receiving messages on a socket an ideal candidate for a multi-threaded process. This way while the server is waiting on a message to arrive, other processes can continue to function.

One of those processes is listening to the serial port on the Arduino for messages from the robot. The robot continuously sends data back to the Arduino, and if desired the Arduino can forward those messages on to the pi. For this the server on the pi polls the serial port for incoming messages and once one is received, it is forward on to the computer over the network.

The serial port on the raspberry pi is considered a shared resource. It can only be written to or read from but not both at the same time. For this a multithreading technique utilizing a mutex to control the ownership of the resource is utilized. In one thread, the server reads from the serial port and in another thread the server writes to the serial port. When writing to the serial port the process checks out a mutex key and prevents the other process wanting to read from the serial port obtaining that resource. The same is done when trying to read from the serial port. Refer to Figure 16 below for the flow chart describing the operation of the server on the Raspberry Pi.
Figure 16. Operation of the message server on the Raspberry Pi.
3.6 Raspberry Pi Streaming Video Data

On the end of the robot arm is a camera used to observe objects near or in the gripper. The camera is connected directly to the raspberry pi where the open source program called RPi-Cam-Web-Interface is running. This program takes each frame of the image captured by the camera and makes it available on a specified Ethernet socket from the raspberry pi. A user can then enter the IP address of the raspberry pi and view the video stream from the camera on a web browser as seen in Figure 17. Software can then be developed to poll that socket on the pi to provide a real-time update on the images in front of the camera on the robot.

![Figure 17. Video streaming from camera on robot over the network to a webpage.](image)

3.7 Robot Control GUI for Windows

To test the wireless functions and the video streaming ability, a sample program was written using C++ on the QT IDE. The Robot Control program has 10 buttons on it in order to control direction of each axis on the robot. There are two text boxes allowing the user to determine the speed and the distance of rotation of each axis. This allows the user to make small or large movements quickly or slowly. In the center of the screen is the video being captured from the camera on the robot. OpenCV libraries are used to capture the images provided by the RPi-Cam-Web-Interface on the raspberry pi and display the stream on the GUI. The image stream will be visible inside the box shown below in Figure 18. In order to put all the axis at a certain angle, one can enter those values in the textbox in the Write Position area. Once the user clicks on send all activated axis will move at the same time. In the event an error occurs, labels representing each servo motor will display the value of the error. Finally, the user can monitor the robots current position in the Read Position area.
Figure 18. Robot Control GUI used to control and monitor the robot's movements.
Section IV: Cost Analysis & Testing
4.1 Cost Analysis

Table 1 shows the cost breakdown of this project. As can be seen in this table, the cost of the Robot Assembly including the servos, makes up approximately 76% of the total costs incurred with this project. Our total cost for this project is $849.91, leaving $150.09 in the budget for future improvements or adjustments.

Table 1: Cost Analysis

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
<th>Manufacturer</th>
<th>Vendor</th>
<th>Quantity</th>
<th>Price/Unit</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX12ABasic</td>
<td>Robot Assembly With Servos</td>
<td>Robot2</td>
<td>GreatCircle</td>
<td>1</td>
<td>$69.00</td>
<td>$69.00</td>
</tr>
<tr>
<td>RD-127A</td>
<td>Driver Channel 12V Power Supply (12V @ 7.7A, 5V @ 2.7A)</td>
<td>Mean Well</td>
<td>TRC Electronics</td>
<td>1</td>
<td>$25.39</td>
<td>$25.39</td>
</tr>
<tr>
<td>Shipping Charges &amp; Tax</td>
<td>Order Number 1/84/10</td>
<td>TRC Electronics</td>
<td>1</td>
<td>$10.54</td>
<td>$10.54</td>
<td></td>
</tr>
<tr>
<td>RHO02R2000FC02</td>
<td>Wound Resistors 12V 2.2 ohm 1%</td>
<td>Victory</td>
<td>Mosher</td>
<td>1</td>
<td>$2.99</td>
<td>$2.99</td>
</tr>
<tr>
<td>RHO02R2000FC02</td>
<td>Wound Resistors - Classic Mount 12V 2.2 ohm 1% MIN 1Watt (12V Class2)</td>
<td>Victory</td>
<td>Mosher</td>
<td>1</td>
<td>$2.99</td>
<td>$2.99</td>
</tr>
<tr>
<td>35-75-5289</td>
<td>Connector for servos</td>
<td>Molex</td>
<td>Newpark</td>
<td>20</td>
<td>$0.23</td>
<td>$4.60</td>
</tr>
<tr>
<td>08-70-1259</td>
<td>Pin for connectors for servos</td>
<td>Molex</td>
<td>Newpark</td>
<td>80</td>
<td>$0.07</td>
<td>$5.60</td>
</tr>
<tr>
<td>SPC15501</td>
<td>MULTICOMP SPC15501 IC &amp; Component Socket; IDC Series, DIP, Socket, 20 Contacts, 2.54 mm, .062 mm, Tin plated Contacts</td>
<td>Multicom</td>
<td>Newpark</td>
<td>1</td>
<td>$0.35</td>
<td>$0.35</td>
</tr>
<tr>
<td>03V05L2412</td>
<td>Orel Bifurc Chip</td>
<td>Texas Instruments</td>
<td>Mosher</td>
<td>3</td>
<td>$0.82</td>
<td>$2.46</td>
</tr>
<tr>
<td>173946</td>
<td>Wire-To-Board Terminal Block 4 pin</td>
<td>Phoenix Contact</td>
<td>Newpark</td>
<td>4</td>
<td>$2.12</td>
<td>$8.48</td>
</tr>
<tr>
<td>211349-3</td>
<td>Wire-To-Board Terminal Black 6 pin</td>
<td>Phoenix Contact</td>
<td>Newpark</td>
<td>1</td>
<td>$0.50</td>
<td>$0.50</td>
</tr>
<tr>
<td>SWH-125-65-G8</td>
<td>5 Position Header Connector 0100° 21x4 Male, Through Hole Tin</td>
<td>Samtec</td>
<td>Newpark</td>
<td>1</td>
<td>$1.73</td>
<td>$1.73</td>
</tr>
<tr>
<td>Shipping Charges - tax</td>
<td>Order Number 028/09</td>
<td>Newpark</td>
<td>1</td>
<td>$0.79</td>
<td>$0.79</td>
<td></td>
</tr>
<tr>
<td>RASPBERRY PI</td>
<td>Raspberry Pi</td>
<td>Raspberry Pi</td>
<td>Amazon</td>
<td>1</td>
<td>$36.00</td>
<td>$36.00</td>
</tr>
<tr>
<td>A0000842</td>
<td>Arduino Due</td>
<td>Arduino</td>
<td>Amazon</td>
<td>1</td>
<td>$16.77</td>
<td>$16.77</td>
</tr>
<tr>
<td>Sensor</td>
<td>Pressure Sensor</td>
<td>Freelance</td>
<td>Mosher</td>
<td>1</td>
<td>$2.99</td>
<td>$2.99</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>JST Female Jumper Male Standoffs</td>
<td>Amazon</td>
<td>Amazon</td>
<td>1</td>
<td>$5.15</td>
<td>$5.15</td>
</tr>
<tr>
<td>BOO/1USB/HIER</td>
<td>12&quot; x 12&quot; x 1/8&quot; Acrylic Sheet for Raspberry Pi Case</td>
<td>Plantronics 2000</td>
<td>Amazon</td>
<td>1</td>
<td>$9.31</td>
<td>$9.31</td>
</tr>
<tr>
<td>BOO/3G/GC8/USB</td>
<td>M3-0.5 Hex Nut</td>
<td>Small Parts</td>
<td>TruFasteners</td>
<td>8</td>
<td>$0.10</td>
<td>$0.80</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$899.91</td>
</tr>
</tbody>
</table>

4.2 Testing

For the testing, the robot could pick up an object that weighed 15oz or 425g. This test shows that the robot can pick up more than the 300g by over 40%. This test was conducted by using motors four and five from a position of 512 to 750. The robot picked up the resting object to the final position.

The next requirements that were proved were the sending of commands and the receiving of data through the wireless interface. The camera was also proven to stream the video wirelessly through the GUI and on a simple webpage that runs off the Raspberry Pi.
We also tested the timing of the longest operation, which is requesting data from all the servos at once. A program was written in C++ to send the command to request all servos data and wait to receive the data. A timer was initiated at the beginning of the send and at the end of the retrieval of data. The time was then written to a text file where it was later loaded into Matlab to be analyzed. This test was done both on IPFW’s network and on a private network in order to rule out network latency issues.

Below in Figure 19 a graph is generated in Matlab showing the time a pc took to request the information to the time the pc received it. On the IPFW network, the average time delay was 2080 milliseconds and the standard deviation was 25.61 milliseconds.

![Round Trip Time Getting all Servo’s Data on Private Network](image)

**Figure 19.** 100 tests of the time delay in getting data from the servos on the IPFW network.
Below in Figure 20 a graph is generated in Matlab showing the time a pc took to request the information to the time the pc received it. On the Private network, the average time delay was 2078 milliseconds and the standard deviation was 40.70 milliseconds.

![Graph showing round trip time getting all servos' data on private network](image)

**Figure 20.** 100 tests of the time delay in getting data from the servos on the IPFW network.

There was not a great difference in the delay between the IPFW network and the private network implying the latency is isolated in the robot control software either in the raspberry pi or the Arduino. In order to receive all the servo’s information, each parameter of the information is requested from each servo. Each request has a small delay to prevent errors in the algorithm. Further research will need to be pursued in order to determine the cause of the large delay and its remedy. Until then, any operations of the robot will need to account for the delay.
References


