Design and Construction of a Half Humanoid Half Rotunda Robot: Rotundaur

Technical Capstone Project
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia, Department of Mechanical and Aerospace Engineering

Team Members:
Emily H. Davenport
Erich Demaree
Russell Hathaway
Alyssa Rorie
Edwin (Win) Sompayrac

Fall Semester 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: Emily H Davenport

Approved: _____ Date **April 22, 2020**
Gavin T. Garner, Associate Professor, Department of Mechanical and Aerospace Engineering

Table of Contents

Executive Summary

Presently, the Mechanical Engineering (ME) Department at the University of Virginia (UVA) only has one student designed robot on display, however it is largely immobile and inaccessible. As robots are an incredible feat of modern mechatronic engineering design, they stand as an advantageous way to interest prospective students in pursuing Mechanical Engineering. The project undertaken in this report sought to create a mobile, user-interactive robot that could potentially serve as a tour-guide for the Mechanical Engineering Building at the University of Virginia. To make the robot more appealing to UVA students in particular, it was aesthetically designed to be a 'Rotundaur" - a half Rotunda and half human robot. The key user interfaces with the robot are a puppet mode, during which the user can move the arms any which way for 15 seconds and then have the robot repeat the motions back, and a controller through which the user can drive the robot's mecanum wheels and open and close its claws to pick up objects. While both of these critical systems worked in the final prototypes, a lack of adequate testing time at the conclusion of the project resulted in the systems being dysfunctional in the final assembled robot. The team recommends future testing on the perfboards, particularly on the analog to digital conversion (ADC) chip, and debugging of the final code loaded to the electrically erasable programmable read-only memory (EEPROM) on the robot to bring Rotundaur to full functionality. The team believes this could be easily achieved in a few days, and future projects focused on the computer science applications that include but are not limited to adding voice, pre-recorded motions, and facial expressions to give an added personality. These additions to the robot could make it a wonderful complement to Departmental tours.

List of Figures

1. Introduction

The overall objective for this capstone design project was to design and build a robot capable of acting as a "tour-guide" of the Mechanical and Aerospace Engineering (MEC) building to interest prospective and current students at the University of Virginia (UVA) in pursuing Mechanical Engineering (ME). In other words, the intent of the robot is to be able to maneuver throughout the second floor of the MEC building at UVA while being directly controlled by a human user through a controller in order to act as a pseudo "tour guide." A reach goal of the project was to add in pre-recorded paths with audio recordings pointing out features of the department on the second floor.

Initially, the team's goal was to build legs for an existing robot torso completed by prior ME capstone students; however, after several conversations with Professor Garner, the team decided against this. The team preferred a larger creative input into the design so as to not be constrained by previous design aesthetics and intent. Instead, the team decided to begin a new robot from scratch to ensure creative freedom with both the functionality in design aesthetics and general purpose of the robot.

After researching current low-cost robotic "toys" available commercially and experimenting with Professor Garner's Meccano robot and Sony Aibo robotic dog for inspiration, the team decided on the Rotundaur theme for the robot. Rotundaur would be half humanoid and half Rotunda robot, with a 3D printed rotunda acting as a chassis for the humanoid torso. The name was derived from both the centaur, ancient Greek mythological creature that is half man half horse, and the Rotunda, the iconic building on the Lawn at UVA. Since the robot is to act as a tour guide to interest current and prospective UVA students in ME, the team desired to maintain a UVA theme in appearance; the Rotunda, being one of the most

well-known symbols of UVA, seemed to be the best way to display this sentiment. In addition, since Thomas Jefferson designed the Rotunda in the neoclassical architectural style of ancient Greece, the team thought this paralleled well with the use of a centaur-derived concept.

To achieve the goal of a robot to interest students in ME, the team decided to design the robot with a focus on an interactive user interface and in conjunction with full mobility capabilities. To meet the user interaction focus, a key design feature of the robot consists of a pseudo-puppet function in which the arms of the robot can be manually moved by the user during recording mode, then repeated back by the robot. To make the robot fully mobile, the team decided to use a mecanum wheel drive-train allowing the robot to move in any direction, including side to side or diagonally, without changing its orientation. A secondary technical goal of the project was to develop pre-recorded routes through the building with audio capabilities that highlight talking points along the way in the building.

The largest constraints to the design considerations were the given budget for the project and the time to complete it. The project budget was originally limited to $600 ($150 per team member) with a timeline of approximately three months (one semester) to design and build the robot. In addition, usable materials also constrained the project. The team attempted to use readily available materials for all possible components of the robot to save money and time. For example, the team chose to 3D print many of the body parts to keep the robot lightweight and the parts easily replaceable. However, the team had to consider printing times and the maximum dimensions of parts to 3D print in the available printers in the Department when designing the robot. For some components, it was unavoidable to order alternative materials; two instances of this included the 8”x8” Aluminum 6061 plate used for the base of the rotunda (robot chassis) and the mecanum wheels. When possible, the team ordered parts from McMaster Carr and Amazon

to save money on shipping costs and time on shipping speed. A final constraint that was paramount during the design process was the aesthetic appeal of the robot. Since it is intended to attract students to the department, the group attempted to design the robot to be attractive and interesting to a diverse range of students at UVA.

2. Background

2.1. Prior Work and Inspiration

The team was inspired to pursue Rotundaur after several weeks of brainstorming and analyzing existing user-oriented, toy-like robots. Two main sources of inspiration were Professor Garner's Meccanoid G15KS Personal Robot, Meccano, and his Sony Aibo ERS-7 dog-like robot. The Meccano robot arrives as a commercially available kit with many thin plastic pieces and hardware that a user assembles into the final robot, pictured below in Figure 1.

Figure 1: Meccanoid G15KS Personal Robot.
The arms of the Meccanoid (Meccano) robot were used for inspiration by the team for the puppet style arms of the team's robot. ("Welcome to Erector by Meccano ® The original inventor brand!" n.d.).

The Meccano robot mainly inspired the team with regards to the arms of the robot. One feature of the Meccano robot is the ability to interact with the user by letting him or her play with its arms and then Meccano can repeat these motions back. The team decided that this feature on

Rotundaur would be a great way to have prospective students interact with the robot and create a sense of awe when the robot repeats back the movement, hopefully intriguing them to pursue Mechanical Engineering and discover how this is possible.

A second robot of inspiration was the Sony Aibo dog like robot. When initially deciding to start a new robot project from scratch, the team thought an animal like robot that is smaller in stature could be an interesting idea. However, after playing with the Aibo dog robot (shown below in Figure 2), the team decided that the key features that made the dog unique were more CS features, like visually tracking user command cards, and overly complicated movements such as rolling over. The team decided that pursuing a similar endeavor could spiral into a greater project that would exceed the one semester time-frame and limited budget.

Figure 2: Sony Aibo ERS-7 dog robot.
Alternate inspiration for the team's robot that helped to eliminate a potentially similar project. ("Sony Aibo ERS-7 | Sony Aibo," n.d.).

Ultimately, after experimenting with these two robots, the team, notably Win, had a revelation for the concept of the Rotundaur. The Rotundaur concept allowed the team the ability to incorporate critical features already decided upon for the robot, such as the "puppet-mode" arms that repeat back user motions, and a mecanum wheel drive train so it can move any direction without changing orientation. In addition, the Rotundaur concept was unique in that it had never been done before and is UVA oriented, which helps fulfill the team's goal of

appealing to prospective and current UVA students.

2.2. Impact of Robotics in Education

In today's technology-dependent society, the importance of encouraging students to pursue science, technology, engineering, and mathematics (STEM) degrees is paramount. Hands-on, experiential learning through robotics, a multidisciplinary field incorporating principles of mechanical engineering, electrical engineering, and computer science, has been shown to be effective in not only teaching STEM concepts, but in engaging students to pursue further STEM education and careers (Garcia, Jimenez, Santos, & Armada, 2007; Mataric, n.d.). Robotics education through extra-curricular activities or after-school programs, such as FIRST Robotics, are incorporated as young as pre-school in some schools, with additional elective classes offered in middle and high school (Skelton, Pang, Yin, Williams, & Zheng, 2010). However, with the high cost associated with funding these teams, the machinery, and materials needed, such robotics experiences are not always available to students before university ("Home," n.d.). As a result, the team wanted Rotundaur to be interesting and available as a "tour guide" to show off to prospective and current UVA students how cool robotics and Mechanical Engineering can be if they haven't been exposed to educational robotics. the team hopes that Rotundaur may one day inspire other students to pursue Mechanical Engineering, even those who may not be confident in their ability to succeed in the program, by showing them Rotundaur's application of many skills gained in ME curriculum at UVA, notably from Mechatronics. In addition, the aesthetic appearance of Rotundaur through its 3D printed parts can show students the impressive 3D computer aided design (CAD) skills learnt in ME. In all, the engineering behind Rotundaur, from the coding, to the CAD design, to the machine design, and to the prototyping, is hidden behind a sleek, fun, and entertaining final design that has the

potential to intrigue and inspire students of all ages to pursue STEM, particularly at UVA.

3. Planning and Budgeting

As a few weeks were lost at the beginning of the semester while the team was still contemplating on the final concept, as soon as the Rotundaur concept was decided upon the team had to get to work while planning simultaneously. To try and keep as on schedule as possible, the team followed Professor Garner's advice of creating a Gantt chart. The original Gantt chart used to track the teams progress on the project can be found in Appendix A. As Professor Garner told the team from the beginning, issues would come up often in every stage of the project, so it was in the team's best interest to fail early and fail often so that the final project would be complete and functioning before the end of the semester. As a result, the Gantt chart was often not followed, especially as the scope of the project adapted throughout the semester, such as when two new group members were added, or unexpected issues came up during prototyping. However, as the team pushed heavily leading up until Thanksgiving, most Mechatronic design and prototyping of the robot were finalized before the break, leaving only aesthetic design of the Rotunda and torso to complete over break. This left only the finalized parts 3D printing and final total robot assembly to wrap up for the final two weeks.

In terms of budgeting, the team tracked the cost of parts purchased for the project throughout the semester through a spreadsheet to compare how much of the $450 original budget had been used. However, as two team members were added with only a few weeks left in the project, the budget increased to $600 which allowed for a more comfortable fiscal position. In addition, budget tracking was made easier by placing part orders through a communal spreadsheet created by Professor Garner. This gave the team a final record of quantities, prices, order dates, and vendors to keep the team accountable. However, the team was very fortunate

that Professor Garner discounted the price associated with any and all 3D printing material or circuit components (including the many fuses and H-bridge chips blown out during prototyping). At the culmination of the project, total expenditures were $490.85, which was under the budget of $750 by $340.15. A full budget spreadsheet and parts list can be found in Appendix B, along with an estimated cost of parts not charged.

4. Design

4.1. Mechanical Design

The biggest challenge of the project was undoubtedly the fundamental mechanical design of the robot. This included the design of all of the joints that are able to be moved by the user or by the servo motors, the arm components connecting these joints, the "torso" of the robot which is the Rotunda, the claws used as the hands, and the base the wheels were mounted to. As the team decided to purchase an already constructed kit of mecanum wheels that included four 12V DC motors to power them, these parts were taken as a "black box" and are not included in this section in terms of mechanical design, aside from mounting of them to the rest of the robot. The completion of the mechanical design of Rotundaur was approached in stages in the order presented above.

4.1.1. Joints

For the rotating joints, several design iterations were undertaken. As a general concept for the arms, the team looked to the Meccano robot. As a result, it was decided upon that the shoulder of each arm would have two servo motors, one to actuate the arm out to the side of the torso, and one to actuate the arm up and down. The arm would also have one more servo motor at the elbow to actuate the forearm up and down.

However, before the team could even begin to design the joints, the servo motors used to

actuate them had to be decided upon in order to know the size and torque constraints. In order to find the specifications needed for these servo motors, estimations of how the highest torque any arm servo motor would have to overcome without stalling were calculated. Since torque is force times radius of the lever arm, the servo motor controlling the raising and lowering of the full arm in the shoulder was considered. Using an estimated arm length of one foot or less and taking the material as 3D printed ABS plastic, the Miuzei DS3218 Digital Servo Motor was decided upon because of its high stall torque for its light weight and small size. While these servos have the ability to travel a full 270° of motion, due to the rotunda-style torso that protrudes as a wide torso, none of the motors are physically able to achieve this full range of motion. The team decided this was suitable since realistically, most human joints cannot travel more than about 180°. In addition, these motors had the ability to be "hacked" so that the team could add a fourth lead from the motor to get feedback on the motor's output voltage in order to know what position is it at all times. This was an important criterion for the team because in order for the robot to repeat back user motions, the feedback signal from the arm motors have to be sent to an analog-to-digital (ADC) convertor chip. Lastly, these servo motors were reasonably priced at only $16 each. Cost was an important factor as the team needed to buy six to eight of these motors for the robot, three for each arm and a potential two more to actuate the head side-to-side and up and down.

Once the servo motors for the arm joints were decided on, the mechanical design of the joints was completed using Solidworks 2019 CAD software. The biggest design implication for the joint was that the team decided it had to be identical for all six motors in the arms to minimize redesigns. In addition, the minimum size was dictated by what could feasibly house the servo entirely so that is was hidden from view to preserve the aesthetics of the Rotundaur appearance

and keep the mystifying effect. The initial joint designed consisted of three separate pieces that assembled with one ball bearing and one servo motor whose output shaft meshed with a 25-tooth servo disc. As will be discussed in the prototyping section, the first joint prototype worked fairly well providing insight into minor mechanical changes to fix interferences in the motion between the three component pieces, to make the bearing fit more exactly as an interference fit, and to make assembly with hardware in tight spaces easier. The largest modification throughout the iterations was that in the final joint two of the original pieces were turned into one continuous piece. Other than these necessary mechanical changes, other modifications were purely aesthetic to take more advantage of the geometry possibilities available by using a 3D printer to construct the joint pieces.

4.1.2. Arm Connections

Once the first joint prototype was completed, the team quickly built an identical skeletal-like upper and lower-arm in order to have a full arm prototype to test with the electronics. Two team members designed different skeletal-like arms, and after finite element analysis (FEA), the stronger design was chosen. One of the few design constraints on the arm was that the team wanted it to be 3D printed out of ABS plastic using the available Dimension uPrint 3D printer, which meant no one piece could exceed the printer's capability of an 8"x8"x8" cube. Another constraint was that the team needed the middle of the arm structure to be hollow to be able to feed wires through it so that they would be hidden from view in the final version of the robot. After the full prototype arm was shown to work with the circuit and code, a more unique and appealing housing for the upper and lower-arms was incorporated into the model. While the changes between the arm prototype one and two seem drastic, they were purely aesthetic in nature as the strength of the arm proved to not be an issue. A prototype chest was also printed

with the second arm prototype, but the team decided to stop the print before completion because the chest was deemed too large.

4.1.3. Rotunda-Style Torso

After two full arm subassemblies (minus the claws) were built and tested with the circuit, the team began the tasks of designing the rotunda torso for the robot and the base the wheels would mount to. The most important decision that had to be made when beginning to design the Rotunda in Solidworks was the overall diameter and height everyone envisioned for Rotundaur. After many talks with Professor Garner, the below mock-up in Figure 3 using the teams finalized arms CAD model was decided upon as a full robot general concept.

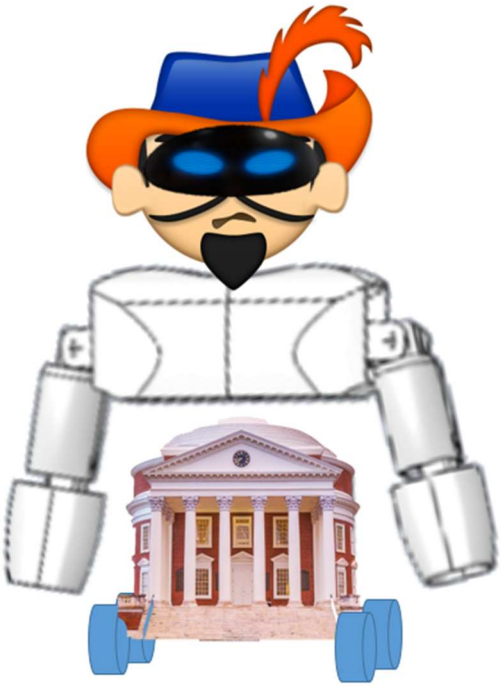


Figure 3: Rotundaur concept art.
Mock-up of the overall robot design as half Rotunda and half Cav-Man using the team's finalized arm CAD model.

After deciding on a maximum of 8" outer diameter for the Rotunda with a 10" height limit, the team broke up the CAD modeling amongst themselves. The team briefly toyed with the idea of using an acrylic cylinder for most of the cylindrical part of the Rotunda, but

ultimately decided existing available ones were too small, too large, or too expensive. The team also thought 3D printing all of the Rotunda would allow for a more cohesive appearance in the final product and allow for more artistic touches, such as choosing the color. The Rotunda CAD work is distinct from the arm subassembly in the fact that it went through several iterations without ever being printed. This was due to the fact that the team wanted to ensure that unlike the chest, where an unfortunate amount of printing material was wasted, no parts of the Rotunda would be printed before a full CAD model was completed, including aesthetic details such as windows, and agreed upon by all team members and Professor Garner. While the final dimensions of the Rotunda could be printed in one print, the team decided to split the two halves of the Rotunda into separate pieces for easy disassembly purposes to access the circuit components intended to mount on the inside. In addition, the team decided to have the aesthetic parts of the Rotunda, such as the stairs and the triangle on top be printed separately and attached using adhesive. A great idea by Professor Garner the team decided to incorporate was to have the columns on the outside of the Rotunda be made out of metal rods rather than 3D printed material to provide structural support. Lastly, the team had to consider how to mount the chest to the Rotunda before printing the final version of either. A circular hole pattern in the interior was decided upon to put four metal rods that would attach all the way through the interior of the Rotunda to the metal base plate. This would provide structural support as well as ensure the chest, Rotunda, and wheel base are all firmly attached.

4.1.4. Claw-Style Hands

The claw design for the hands was an interesting challenge, as it was undertaken once the rest of the arm design had already been tested and finalized with the electronics. As a result, the space allowed for all the components was limited by prior arm design. Second, materials and

mechanics for the claws were limited due to the necessity to be lightweight and rapidly available to be added onto the arms. To ensure the design fit into the space, design of the claw mechanism was undertaken completely in Solidworks to insure a proper fit and movement constraints. In order to allow for rapid prototyping and intricate fabrication of custom three-sided racks and gears, the parts were decided to be 3D printed aside from the motors used to actuate the claws.

The available working space to create a claw system inside the forearm was limited to a cross section of 2 by 2 inches and a depth of 4 inches. Research into available motors to drive the claw lead to the conclusion a linear actuator would be the best option, as they are generally smaller and lighter than most DC motors or rotary servos. Two RC linear servo motors weighing 15 grams, with dimensions measuring 15mm x 21.5mm x 47mm, a 20mm stroke, and a peak push and pull efficiency of 20 Newtons was sourced from Actuanix at a cost of $70 each, shown in Figure 4

Figure 4: PQ12-R 100:1 Ratio RC Linear Servo Motor. Motor used to actuate the claws.

Knowing the size and stroke length of the actuator provided a starting point for the claw design in Solidworks. A solid body model of the actuator was created and positioned such that the center of the extending arm would line up with the center point of the forearm opening. A mounting structure to support the actuator inside the forearm was extruded from the interior walls, with a location to secure the included U-bracket and take advantage of the grove along the

front to hold the actuator securely in place during operation (see Figure 5 below).

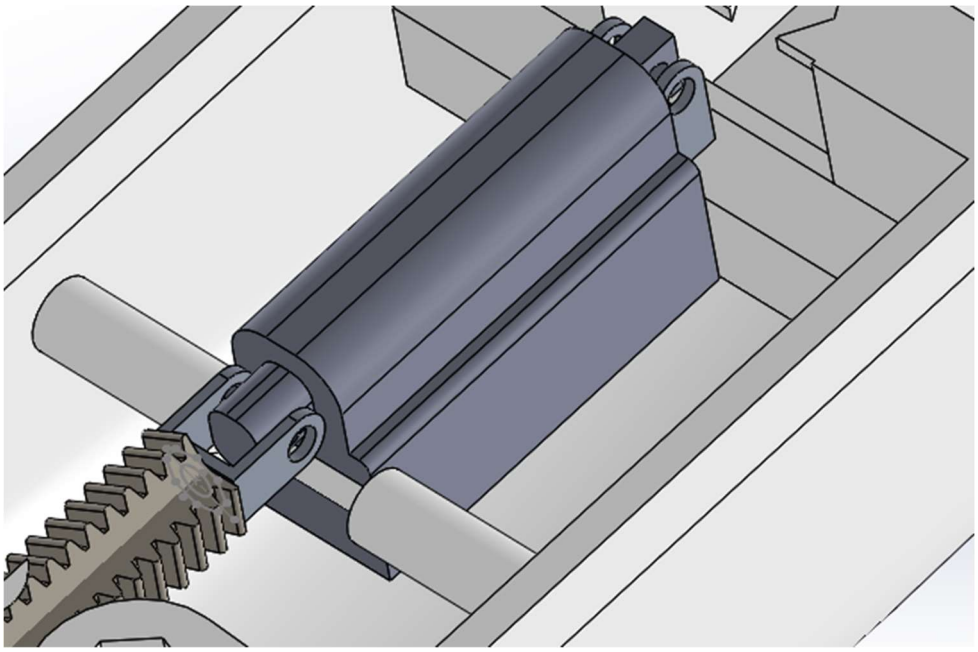


Figure 5: Linear Servo modeled in Solidworks. Servo positioned within the forearm with support hardware and custom three-sided rack.

To take advantage of the linear motion of the servos, a rack and pinion design to actuate the movement of the claws was used. With the team's decision for a 3-clawed hand style, a 0.7-inch gear diameter allowed for inclusion of a rack and room for the claw fingers to move without interferences. The 0.7-inch diameter would also allow the claw to be integrated directly onto the gear, taking roughly half the circumference of the gear, while still taking advantage of the full 20mm stroke of the actuator. The gear design consisted of 26 teeth with a module of 0.027 inches and a 20-degree contact angle (See figure 6 below).

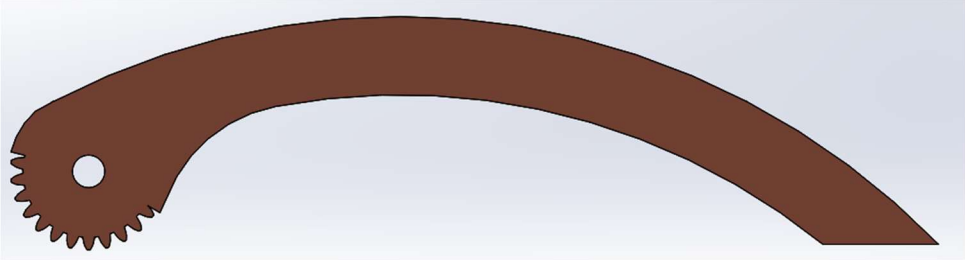


Figure 6: Side view of one piece of the three identical used to form one claw. The gear aligns with the three-sides rack mounted to the linear servo to open and close the claws.

A triangular rack was designed with matching values to the gear, with each gear face oriented 120 degrees from the other, and measuring 25mm in length to ensure full contact is maintained with the claw's gear at all extension positions (See Figure 7 below).

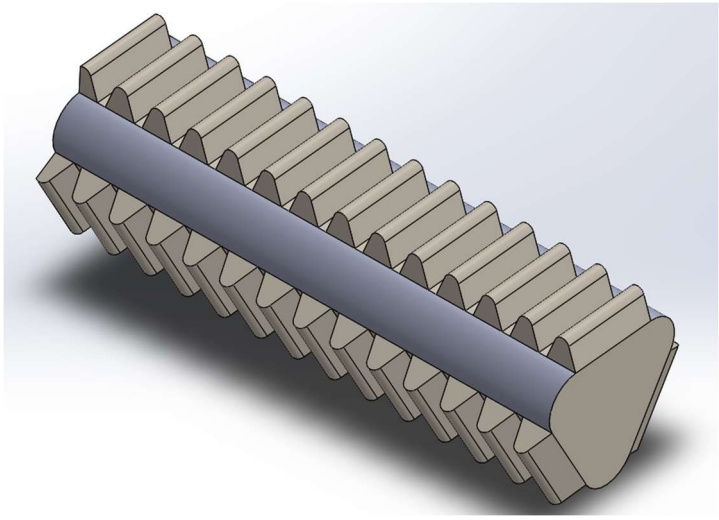


Figure 7: Isometric view of the three-sided rack. Three-sided rack which mates to the matching gears on the claws to actuate them open and closed.

The rack was centered linearly with the actuator, connected to the actuator with the included hardware, and held in-line by the three claws. As the linear actuator extended and retracted, the teeth on the rack would symmetrically open or close the claws.

Once the actuation for the claws was designed, a method of holding them in place was constructed. The team opted to not use bearings for the rotational axis of the claw due to the increased weight it would add to the end of the arms, the slow speed of operation, and the lack of

need to grip with significant force. Instead, the claw would rotate around its mounting screws. A cutout was made in Solidworks at the end of the forearm to allow the claws to open to their full capable range. Finally, holes of 0.06-inch diameter were modeled to allow access for a screwdriver to easily reach into the forearm for assembly. Figure 8 shows the fully constructed forearm-claw subassembly modeled in Solidworks before the forearms, geared claws, and racks were 3D printed.

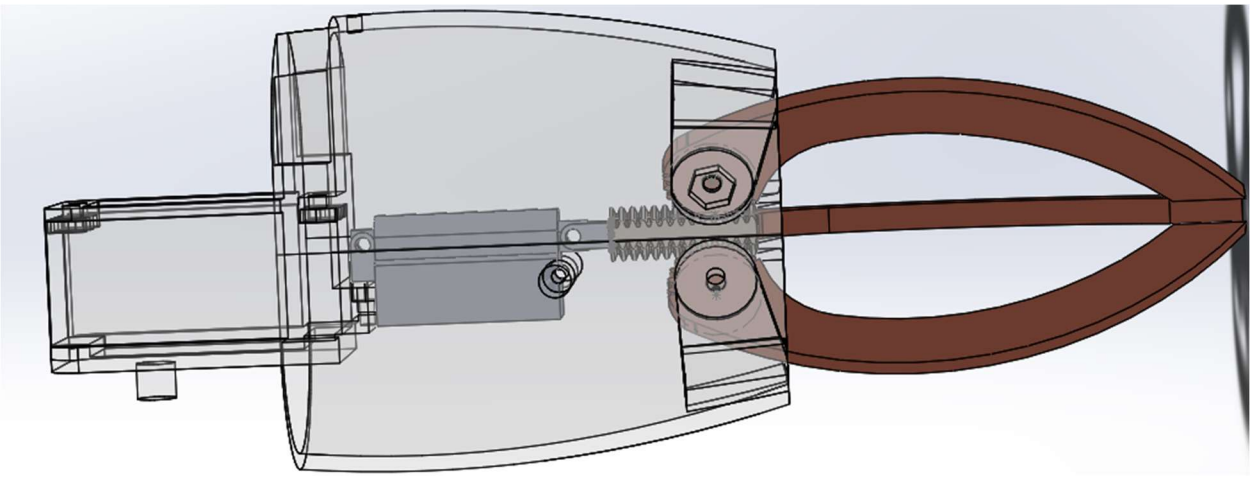


Figure 8: Completed forearm sub-assembly. Forearm sub-assembly containing the forearm, linear servo motors, three-sided rack, and geared claws.

4.1.5. Wheel Base

The last purely mechanical design involved in the robot was the design of the base for the mecanum wheels to attach to. An initial prototype rectangular base was designed in Solidworks then laser cut out of ¼" thick acrylic in order to be able to mount and test out the circuit and code to control the wheels. This base was never intended to be permanent as acrylic can shatter easily, but worked well in terms of being able to mount the prototype breadboard, a battery pack, and other electronics components to while testing driving the wheels. Afterwards, it was decided upon to construct the base out of an aluminum sheet so that it would be sturdy against shattering and strong enough to hold the rest of the robot. The team wanted the base to be hidden

underneath the Rotunda so that only the wheels were visible, focusing the viewer's attention on the Rotunda itself. As a result, the base was designed to be “quadcopter” style - a circle with the diameter matching the Rotunda with four rectangular corners coming off each with four 9/64” holes for mounting to the wheels. The final base was cut out of an 8”x8” metal sheet on the waterjet cutter.

4.2. Electrical Design

The electrical design of Rotundaur had two main subsystems, the arms and the wheels. For the circuit prototyping, the Mechatronics Experiment Board (MEB), a portable breadboard, and an External Power Supply capable of supplying up to +14V were used. A Tektronix Oscilloscope was used for debugging hardware versus code issues. Spin programming was used for all code to control a Parallax Propeller chip powered off of 3.3V. Much of the motor and analog-to-digital converter (ADC) chip testing code originated from the Mechatronics course textbook written by Professor Garner. The arms circuit was constructed on the MEB simultaneously as the joint and arm CAD models were designed in Solidworks and then 3D printed to test with the circuit and software. The wheels circuit was constructed on the portable breadboard simultaneously as the torso and Rotunda were designed in Solidworks. Testing for the arms circuit was done on the MEB and the wheels circuit on the portable breadboard and temporary acrylic base before moving to perf boards to ensure the circuit and software were perfected.

As shown below in Figure 9, the arm circuit consists of the feedback of the six servo motors going into a 12-bit ADC chip that converts the input signal received from each motor into a 12-bit number between 0 and 4095. The arm servo motors are all powered off of +5V. In addition, the arms circuit for testing on the MEB contained the Radio Frequency (RF) module to

transmit signal from the NES Controller wired to the adaptor on the MEB board. This allowed for the controller to transmit "wirelessly" to the wheels for testing so that the base and wheels were not limited by being tethered to the controller. This will be discussed further in the controller section below.

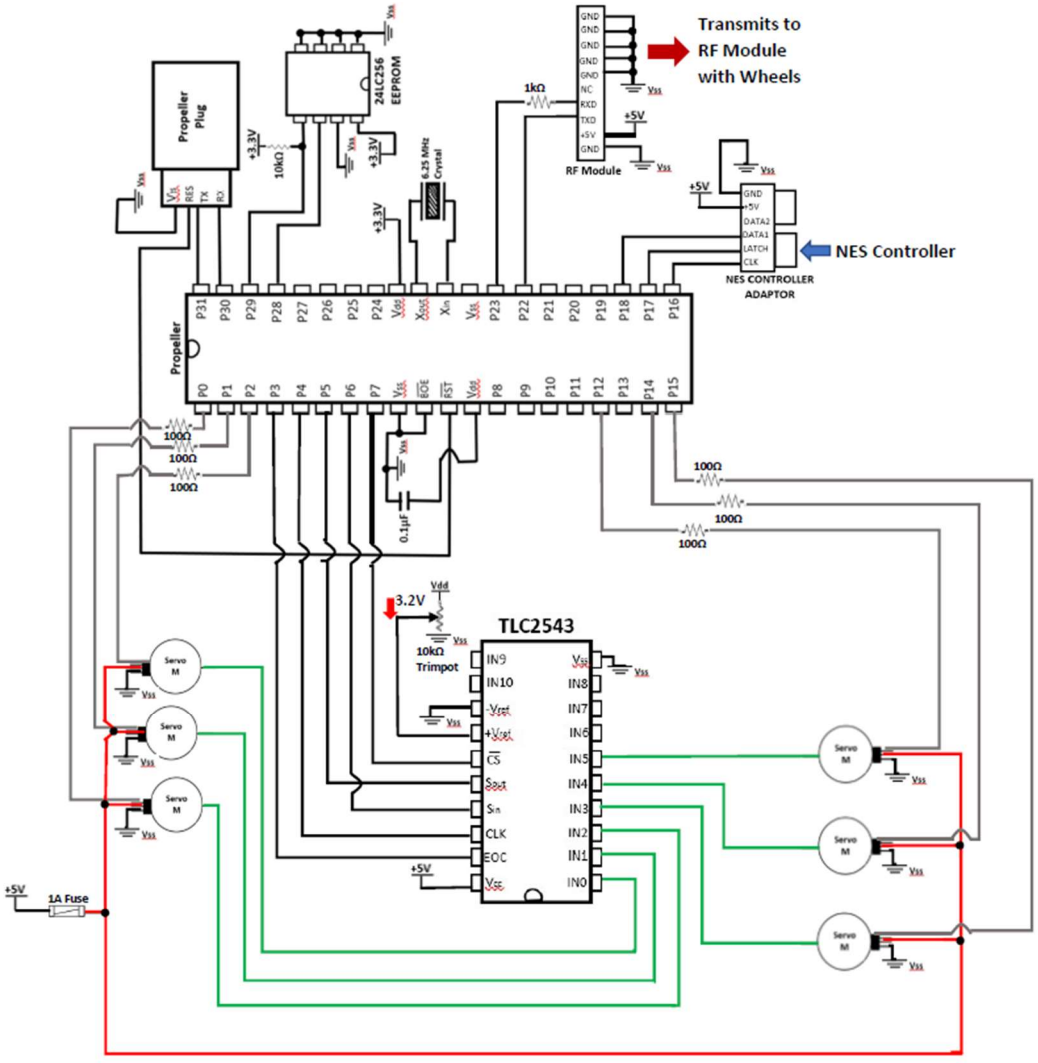


Figure 9: Arms circuit and wiring diagram.
Circuit diagram for the Propeller chip, 6 arm servo motors, the ADC chip, and wheels controller.

of the analog to digital conversion. The common ground for the entire circuit was used as the $-V_{ref}$. The ADC chip reads each channel containing a servo feedback input one after the

other then sends the digital number to the propellor chip. The code sample below in Figure 10 has the Propeller chip store the sampled data from each servo motor, then converts it to the analog signal needed for the servo motor to repeat the motion, then tells the servo to go to that corresponding position. The pin numbers for the ADC chip and the six servo motors were defined in the constants (CON) section, and a separate ADC method used. The Servo32v7 object is used to run all six arm servos off of a single cog. To convert the stored digital number back to an analog signal to send to the servos, a scaling factor was calculated as the range of the servos divided by the number of bits. This scaling factor was multiplied by the stored digital number from the ADC then divided by 100 to convert it to the Servo32v7 range which is between 500 and 2500 only (whereas the motor's default is between 50_000 and 250_000). A waitcnt for 20ms was added to match the sampling rate of the 6 channels by the ADC (set with the endcnt variable in the sampling loop) and allow the motors sufficient time to move.

```
PUB Main | i,j, samples, endTime

  s.Start                       'StartServo32 obj on cog1
  s.Ramp                        'Cog2; Using SetRamp method requires another cog for the Ramp object called WITHIN servo32

  outa[cs]~~                    'Start the ADC's chip select pin High
  ADC(0)                        'Set the initial channel to read the ADC on as 0 (Motor0's ouptut)
  samples:= 749                 'Set number of smaples you want to get of position data
    '750 total samples * 20ms = 15s of sampled data

  repeat i from 0 to samples
    endTime:=cnt+(2_000_000)      ' 20ms adjusted to Servo32 code to allow motors time to move

    ADCdata0:=ADC(1)             'Set ADC multiplexer to perform NEXT conversion on Channel 1 (Motor 1's)
    position_data0[i]:=ADCdata0

    ADCdata1:=ADC(2)             'Set ADC multiplexer to perform next conversion on Channel 2 (Motor 2's)
    position_data1[i]:=ADCdata1

    ADCdata2:=ADC(3)             'Set ADC multiplexer to perform next conversion on Channel 3 (Motor 3's)
    position_data2[i]:=ADCdata2

    ADCdata3:=ADC(4)             'Set ADC multiplexer to perform NEXT conversion on Channel 4 (Motor4's)
    position_data3[i]:=ADCdata3

    ADCdata4:=ADC(5)             'Set ADC multiplexer to perform next conversion on Channel 5 (Motor5's)
    position_data4[i]:=ADCdata4

    ADCdata5:=ADC(0)             'Set ADC multiplexer to perform next conversion on Channel 0 (Motor 0's)
    position_data5[i]:=ADCdata5

    waitcnt(endTime)            'Wait for exactly 20ms every sampling loop (adjusted period for Servo32)
```

Figure 10: Spin code for the arms of the Rotundaur. (Code and explanation continued below).

```
repeat i from 0 to samples
  s.Set(Servo0, (position_data0[i]*49+50_000)/100)     'Set(MotorPin#,PulseWidth)
                                'scale back to between 50_000 and 250_000, then convert to servo32 range by div. 100
  s.Set(Servo1, (position_data1[i]*49+50_000)/100)
  s.Set(Servo2, (position_data2[i]*49+50_000)/100)
  s.Set(Servo3, (position_data3[i]*49+50_000)/100)
  s.Set(Servo4, (position_data4[i]*49+50_000)/100)
  s.Set(Servo5, (position_data5[i]*49+50_000)/100)
  waitcnt(clkfreq/50+cnt)                           '20 ms delay to match ADC sampling rate & so motors have time to move
```

Figure 10 (cont.): The motors' feedback is sent into the ADC channels, then each sampled in turn, then sent to the Propeller chip as a digital number, then converted back to an analog signal, and sent back to each motor in turn, and each motor moves to the corresponding position through servo32.

After testing was undertaken to ensure the arms were repeating back the exact motions performed by the user, it was found that the Set command from Servo32 had a bug that caused the motors to not move at all, or be off in their final position if it was too drastic of a change in position. To correct this, the SetRamp command was employed from the Servo32 object as well. After the Set command for the motor, the SetRamp command was used to dictate the amount of time the servo should take to complete the motion from its current position to the new position. The final arm code using the SetRamp command can be seen in Appendix C.3. SetRamp required the use of an additional cog for the "Servo32_Ramp_v2" object called within Servo32 but this was not an issue. Since the time between ADC samples for each motor is 20ms, 20ms was used. The waitcnt at the end of the loop was redundant now, so it was removed. The SetRamp command was found to alleviate the motors not moving at all between positions, but occasionally the motors still did not move back to their initial position, usually when the final position was near the minimum position of the motor.

For the wheel circuit (shown below in Figure 11, each mecanum wheel has a DC motor attached that is powered off of +5V and connected to an H-bridge chip powered off of +12V that can switch the direction of motion to CW (0) or CCW (1). The Moebeius 80mm Mecanum Wheel Robot Kit was purchased from Amazon to use for the team's project. The team decided to purchase this kit rather than buy DC motors and mecanum wheels separately because it was

less expensive and contained the approximate size wheels that fit Rotundaur's scale. The team designed and built a base rather than use the one that came with the kit to suit the team's aesthetic needs. The team authored all code used to drive the wheels as well. An RF module within the wheel's circuit received data from the NES controller via another RF module on a separate Propeller chip and circuit containing the controller. During initial testing, the NES controller was connected through the adapter shown in Figure 9 directly to the robot. After this was proven to work, the wireless receiver was added.

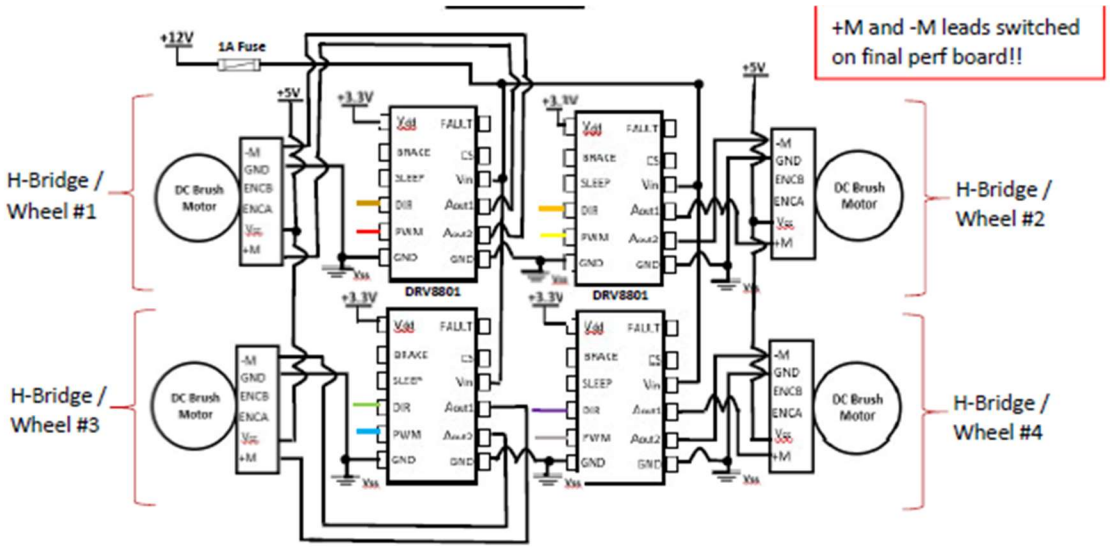


Figure 11: Wheels circuit/wiring diagram.
Each mecanum wheel has an attached servo powered by an H-Bridge chip that can tell it to rotate CW or CCW. A wireless RC receiver receives signals from the NES controller connected to the MEB to control how the robot moves.

The mecanum wheels used to drive the robot are unique in that they allow for a holonomic drive - the robot can move in any direction without changing the orientation of the front of the robot. This is possible because the mecanum wheels are made of rollers oriented at 45° to the motor shaft, as shown below in Figure 12. This includes forwards, backwards,

strafing left and right, diagonally any way, and rotating CW or CCW.

Figure 12: 80mm mecanum wheel robot kit by Moibeius.
This robot kit was purchased by the team but only the mecanum wheels, DC Motors, and mounting hardware were used, the team designed an independent aluminum base.
("Amazon.com: Moebius 4WD 80mm Mecanum Wheel Robot Car Chassis Kit with DC 12V Encoder Motor for Arduino Raspberry Pi DIY Project STEM Toy: Industrial & Scientific," n.d.).

It was always desired that Rotundaur would be able to move around instead of remaining stationary like its other robotic counterparts, and mecanum wheels were an ideal drive train as Rotundaur could move in all directions, and it's seamless ability to strafe in any direction and turn on a dime brings awe to spectators that are not used to that type of motion. In order to achieve these motions, the wheels must be mounted in a rectangular pattern and exactly symmetrically since the overall robot motion is determined by basic vector addition, as seen in Appendix C.1. A diagram of the direction each of the four individual wheels are driven to create each of these motions is shown below in Figure 10. For the purposes of the robot, the full speed of the wheels was excessive, so it was scaled back to 25% power.

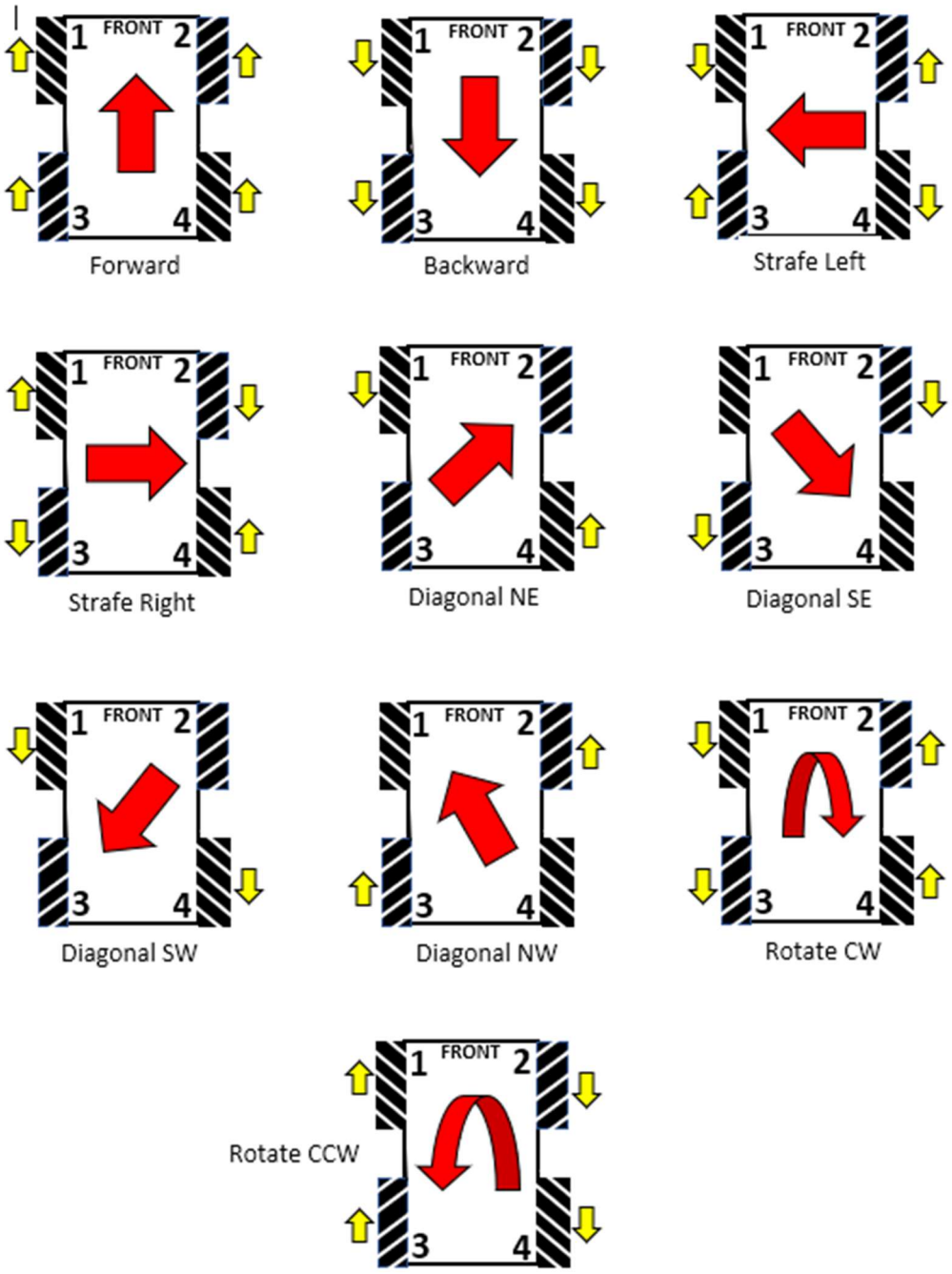


Figure 13: Individual mecanum wheel driven direction and overall robot motion diagram. The individual wheels are labeled 1-4 going CW around the robot. The referenced front of the robot is labeled.

4.3. Claw Design

When claws were added to the mechanical design of the robot, RC linear servo motors were chosen to actuate the three fingers of the claw along a three-sided rack within the arm. The

circuit for the RC linear servos is nearly identical to the DC servos used in the arms, except they do not have the "hacked" feedback wire. The three leads are connected to ground, +5V power, and to one pin on the propeller chip. This is shown in the full propeller circuit diagram in appendix C.3. The software to control the motors is also nearly identical to the arm DC servos, except the pulse width range is between 100_000 and 200_000. However, Servo32 was still able to be used to control the linear servos and no ramping functionality was found to be needed.

To achieve precise motion of each claw so that a user could pick something up, it was decided to incorporate the motion of the linear servos on the controller used with the wheel. The control of the claws with the controller is discussed below in the Controller design section. When the buttons used to open and close the right or left claw are pressed, a small value is added or subtracted from the pulse width value used in the Set command to move the motor to the position. Figure 14 shows a snippet of this code. As a result, a discrete number of steps can be achieved between the "fully open" and "fully closed" positions of each claw.

```
%1001_1111:        'SELECT button & B pressed (Left Claw open)
  positionL:=(positionL-500)          #>100_000
  s.Set(LinServoL, positionL/100)               'Use Servo32 object to run LinServos
  waitcnt(clkfreq/100+cnt)

%0101_1111:        'SELECT button & A pressed (Left Claw close)
  positionL:=(positionL+500)          <#200_000
  s.Set(LinServoL, positionL/100)
  waitcnt(clkfreq/100+cnt)


%1010_1111:        'START button & B pressed (Right Claw open)
  positionR:=(positionR-500)          #>100_000
  s.Set(LinServoR, positionR/100)
  waitcnt(clkfreq/100+cnt)


%0110_1111:        'START button & A pressed RLight Claw close)
  positionR:=(positionR+500)          <#200_000
  s.Set(LinServoR, positionR/100)
  waitcnt(clkfreq/100+cnt)
```

Figure 14: Section of controller code. A section of the controller method running a case to open and close arms.

4.4. Controller Design

A Nintendo Entertainment System (NES) controller as shown below in Figure 15 was used to control the robot. This controller was chosen by the team because members already knew how to work with it from previous coursework, it was readily available, and free to use.

Figure 15: NES Controller used to control Rotundaur.
The dual axis pad, A, and the B button can be used to move the robot.

The Spin code excerpt in Appendix C.2. details how the buttons are read into the NES adaptor by the Propeller chip as an 8-bit binary number corresponding to each movement and then received by the RF module to power the wheels' motors. Table I below details what buttons correspond to which robot movements. If none of these combinations are pressed, the robot stops all movement. The final code for the circuit used on the separate propeller chip off the robot to transmit the NES controller data to the robot is shown in Appendix C.2.

Button(s) Pressed	Robot Movement
Up	Forwards
Down	Backwards
Left	Strafe Left
Right	Strafe Right
Up & Right	Diagonal North-East
Up & Left	Diagonal North-West
Down & Right	Diagonal South-East
Down & Left	Diagonal South-West
B	Rotate CCW
A	Rotate CW
Select & B	Open Left Claw
Select & A	Close Left Claw
Start & B	Open Right Claw
Start & A	Close Right Claw

Table I: NES controller button combinations and corresponding robot movements.

4.5. Aesthetic Design

The use of the Dimension uPrint 3D Printer for fabrication of most parts allowed the team to print parts of any shape desired, as long as it could be created in Solidworks and fit within an 8" cube. This was crucial for the design of the Rotunda chassis, as all of the complex features such as the dome, columns, stairs, and small aesthetics such as windows could be designed in Solidworks and then printed. While many of the prototyping was done with blocky, classic robotic looking ABS parts, Professor Garner inspired the team to use the printer to their advantage and to design more creative, realistic Rotunda-looking pieces. There was no point in 3D printing parts that the team could easily buy on the internet. With this in mind, the final design of the parts for the robot are more reminiscent of the Rotunda. In addition, the arm assembly features ellipsoid shapes for the forearm and bicep, mimicking that of a real human. The torso also attempts to incorporate curves to imitate shoulders and pectoral muscles.

The interior of the Rotunda proved to be useful not only for aesthetics, but for mounting copious amounts of wires and circuits that needed to be incorporated for the arms and wheels,

yet hidden from view to preserve the appearance. Knowing these components may need to be accessed for modifications or replacement, the Rotunda was designed in multiple independent parts to be printed and then bolted together. The Rotunda was built in two halves cut down the x-z plane creating two 3D semicircles. To bolt the two halves together, tabs were created on the inside of each Rotunda half in order to put bolts through. This eliminated the visibility of the bolts, giving the illusion of the Rotunda as one part.

In the final week of the project, LEDs were decided to be added to give Rotundaur one final enchantment. Code was tested with the LEDs and recesses with mounting holes incorporated into the final printed arms for them, however the lack of testing time at the final assembly of the project prevented the LEDs from coming to fruition on the final Rotundaur.

5. Final Design

The overarching theme, as previously stated, was to design a tour guide robot, that maintained a UVA theme. The constraints of 3D printing in an 8” x 8” x 8” cube proved to be difficult, especially with large parts such as the torso and rotunda, and having to fit the lithium ion phosphate battery pack on the inside. In addition, parts needed to be able to be easily assembled and disassembled for access to the wires and circuitry. To maintain the UVA theme and aesthetic appeal of the robot, individual parts of the robot were printed with specified colors. The arms and torso were printed in blue, while the rotunda chassis was printed in orange with white aesthetic triangles and stairs. As a 3D printed, approximately 12” robot would be fragile, key fixtures on the shoulder were thickened, and 8.5” x .375” threaded rods with nuts were used to provide total structure as well as provide a manner to attach the 3D printed material to the aluminum base. Figure 16 depicts a 3D rendering of the final Rotundaur robot concept.

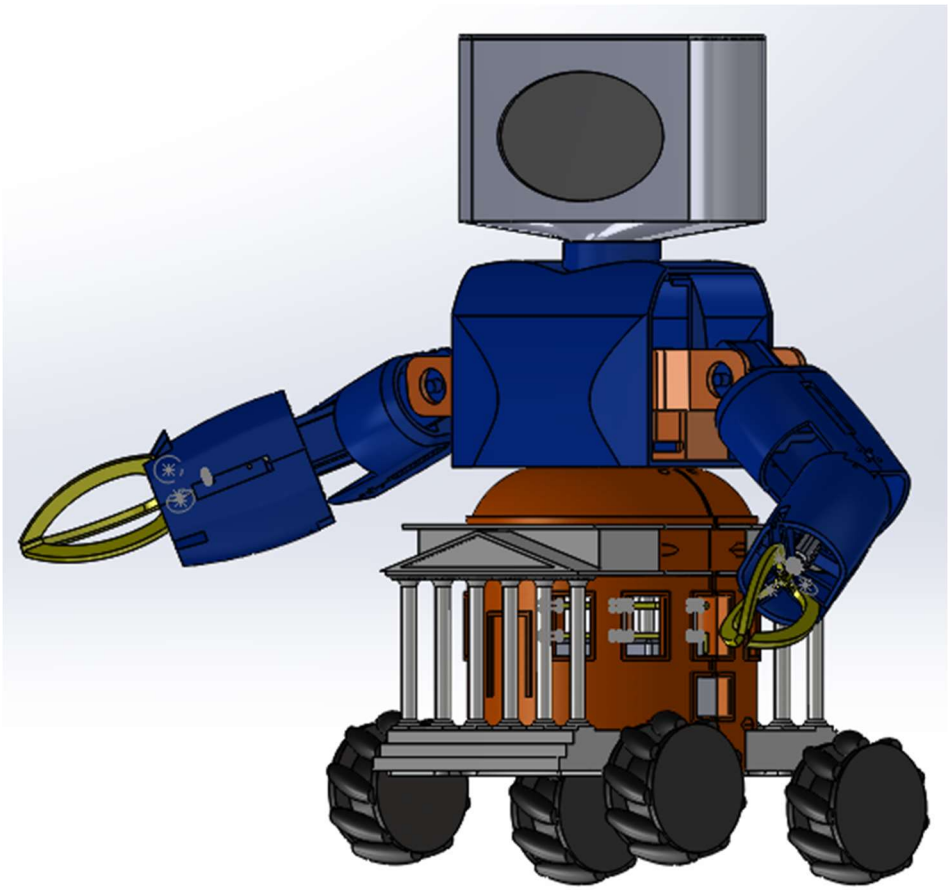


Figure 16: Final concept of Rotundaur Robot. Color scheme for printed material. Interior not currently shown.

To provide support to the aesthetics of the robot, it was decided to use .375” aluminum rods cut to 4” in length. The polished versions of these rods help set the aesthetics apart from the chassis of the robot. While the yellow claws do not follow the UVA theme, they serve as a focal point on Rotundaur and being a physical representation of attitude. This provides a manner of providing multiple dimensions to Rotunudaur’s personality as it has a cute, but aggressive appearance.

The interior of the robot provides room to store the battery; however, as the battery was not a key factor in the original design, it became important to design a manner so as to secure the battery, and provide support to internal circuits. Figure 17 depicts the battery brace with a

cantilevered shelf to attach perfboards.

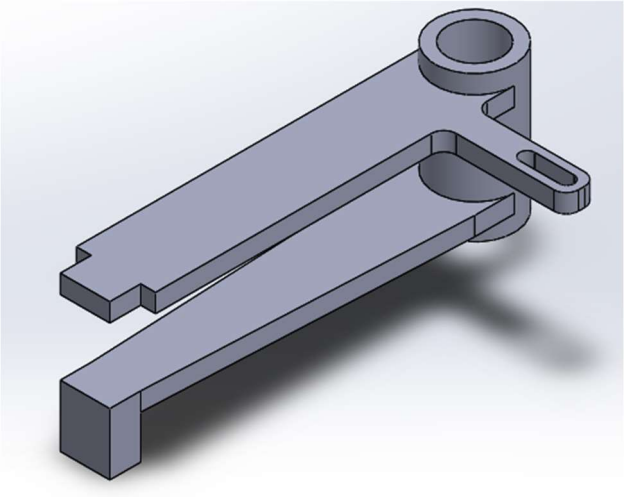


Figure 17: Right Battery Bracket. Bottom section secures the battery to the base of the Robot; while the cylinder, located in top right, slides over the .375" threaded rod, and secures the perf board by the slot pictured right side.

Two switches were also incorporated into the final design, one to shut on and off the power to the wheel's motors, and one to switch on to start puppet mode. They are located on the bottom of the base plate on the back of the Rotunda, respectively.

6. Prototyping

In order to ensure the Rotundaur project was feasible, a prototyping phase was necessary. During the prototyping phase of the design, key mechanisms were set the focus as small projects to work together. Most notably, the entire arm assembly was broken into three areas of focus as each sub-assembly built upon a part of the previous assembly. These sub-assemblies were: Shoulder Joint, Arm Sub-Assembly, and Claw Sub-Assembly. The wheels and controller assembly served as a separate area of focus as it incorporated the wheel and motor aspects, and the wireless control as two entities.

6.1. Shoulder Joint

As previously stated, the Meccano Robot joints served as a basis for understanding how

to design the Rotundaur's arm movements. The initial designs focused on the shoulder joint and copying the joint in different configurations to provide rotations in different planes for the various joint movements. Figure 18 below to the left, identifies the first joint prototype with three independently designed parts that had to be assembled with hardware in white and the servo motor with connector hub in grey. Figure 18 below to the right depicts a variation of connecting two joints together, such as in the shoulder, enabling multiple planes of movement.

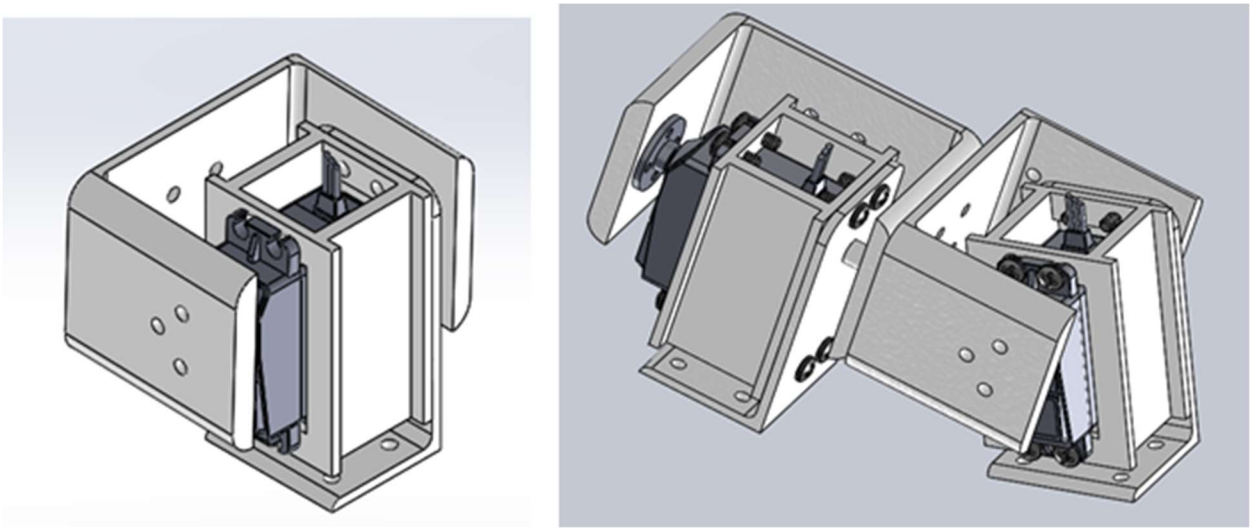


Figure 18: Pictured left. Right Battery Bracket Single prototype of an assembled joint. Pictured right. Two joint prototypes combined. Joint prototypes enabled movement of either the small C-device or motor itself about the axis of rotation in the motor.

From this prototype, the team learned what aspects of the designs worked well, and what aspects needed to be fixed. In general, the motor mount, pictured in Fig. 18 as a small I-beam style box, held the motor well and served as a great platform to base the remaining parts. One of the two largest issues with the prototype design surrounded how to attach the motor mount to the rest of the robot. In the first iteration, the motor mount was cantilevered from a .1" thick hanger. This enabled small vibrations to be amplified throughout the joint and was very unstable as more

weight was attached. In the second iteration, the mount hanger, pictured in Figure 19 right, was extended and made into a continuous “J” shape to increase stability.

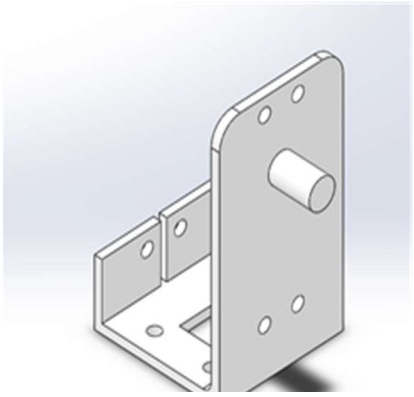
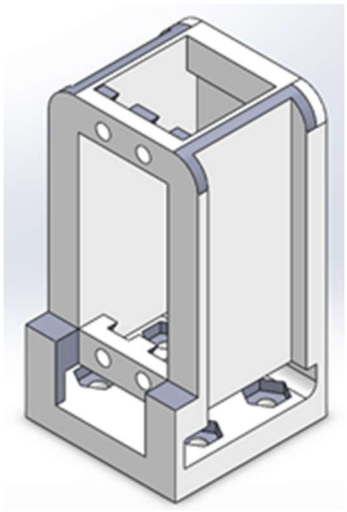


Figure 19 (above): “J” shaped mount hanger.
Figure 20 (below): Finalized motor mount.



This minor change helped dampen undesired vibrations from the motor and made the joint more rigid. After testing a completed arm, see 6.2 section, it was determined that a redesign was needed for the joint assemblies. The purpose of the redesign was to reduce the amount of hardware necessary to construct the robot arm, remove additional weight from the hardware, and to simplify areas that could be made into single pieces. In doing the redesign, the motor mount and hanger became a single piece with recesses for the 6-32 hex-nuts, as seen right in figure 20. The recesses added some material to the overall mount; however, they provided a much-needed area to secure the hex-nuts in the small spaces. Also, the bottom of the hanger was thickened to provide extra material in case an arm was impacted by a user.

Lastly, by redesigning the C-bracket, as seen in Figs 18, the interferences caused by needed hardware and excess movement caused by the tolerancing in 3D prints could be removed. To begin redesigning this part, the two C-brackets were brought together as a single part, in a manner which minimizes the amount of hardware needed in a joint. This can be seen in

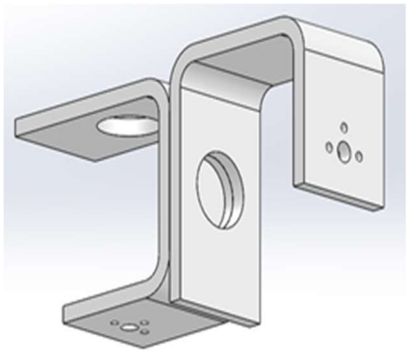


Figure 21: Updated C-bracket.

Fig. 21 to the below. In this design, the only hardware needed would be R188 ball bearings, and the round-hub motor connector with correlated hardware. While the design would work, the aesthetics would need reworking to make it a sleeker model. To finalize the redesign, the entire C-bracket was combined in Solidworks and minor extrusions and cuts were made to make a more aesthetically pleasing unit. This can be seen in Fig. 22 right. This final model enabled the motor and its bracket to be secured with the aforementioned ball-bearings and hardware while still enabling the arm assembly to be connected to the outside easily, as previously mentioned in the Final Design (section 5), and described below in section 6.2, Arm Subassembly.

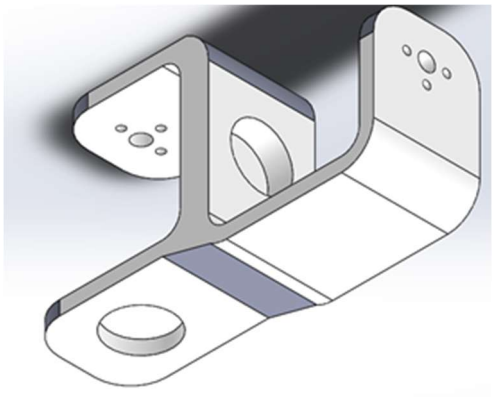


Figure 22: Finalized Shoulder rotator. Connects the shoulder mount to the arms.

6.2. Arm Subassembly

The first design for the arm sub-assembly was intended to mimic a skeletal structure which could later have muscular definition added to with additional pieces. In the initial design, the upper arm and forearm joints were connected via a single beam to the base of each joint subassembly in the shoulder and elbow. The forearm prototype was based off of the hole pattern in the original C-bracket, as mentioned above in section 6.1. The original arm subassembly can be seen in Fig. 23 seen below. This figure depicts a completed skeletal assembly with the base of the shoulder seen at the left and fore-arm section seen in bottom right.

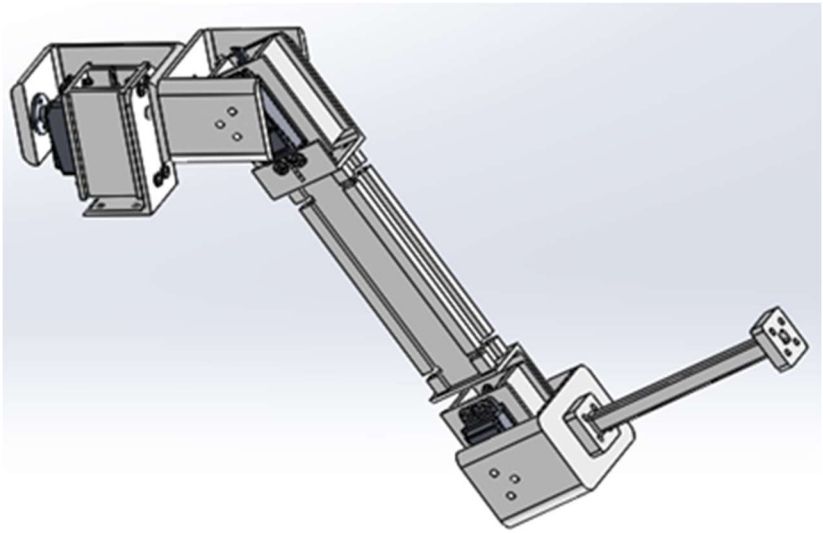


Figure 23. Initial arm prototype. A rudimentary arm skeletal structure with combination between arm joint prototypes.

After initial testing of the arm, the team decided that the amount of hardware needed to complete a second arm, and overall size of the first arm, measuring approximately 18” from shoulder to tip of forearm, required a major redesign. In addition, the skeletal design with added on muscular pieces was redundant, and only the muscular exterior shells were needed in addition to the joints. To maintain the ability to print the final robot in the available 3D printers, a maximum total arm length of nearly 10” was decided. To meet this size constraint, each arm section would need to be approximately 5” long. Instead of each arm section having multiple parts like the earlier prototype, the team decided to focus redesign efforts on having a single arm unit to house the motor. This proved difficult as it took nearly 20 hours to finalize a design for the upper arm. Figure 24 below depicts the final upper arm prototype.

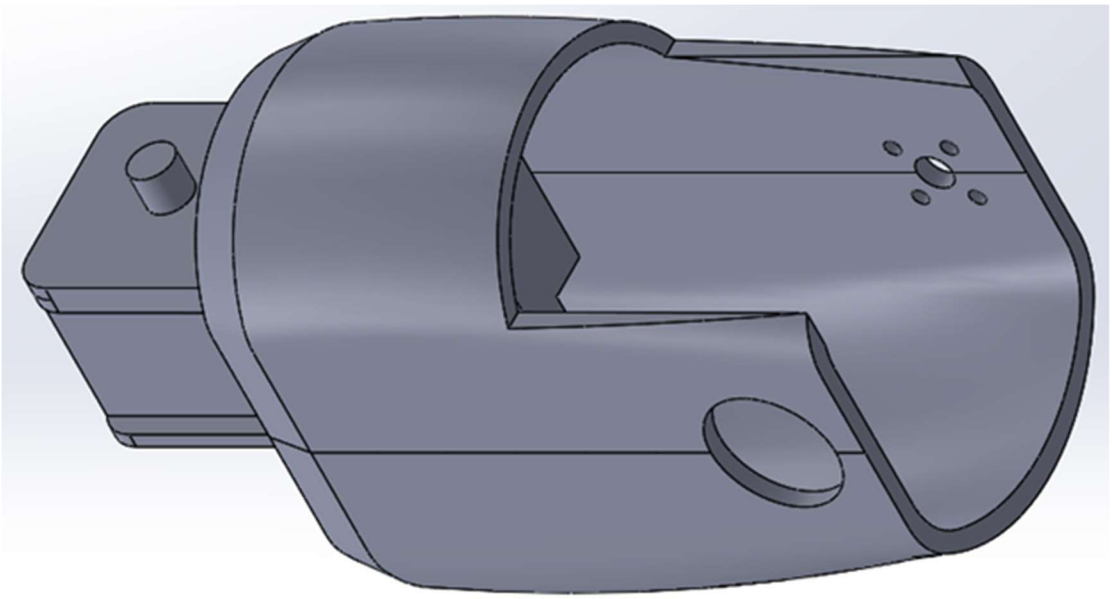


Figure 24: Upper arm final prototype. The upper arm of the robot with a muscular bicep feature for aesthetics. The large, circular hole is for a R188 ball bearing and smaller holes are for motor hub connections

To save on time, it was decided to make the upper and forearm similar parts. Figure 25 below depicts the fore-arm prior to claw design.

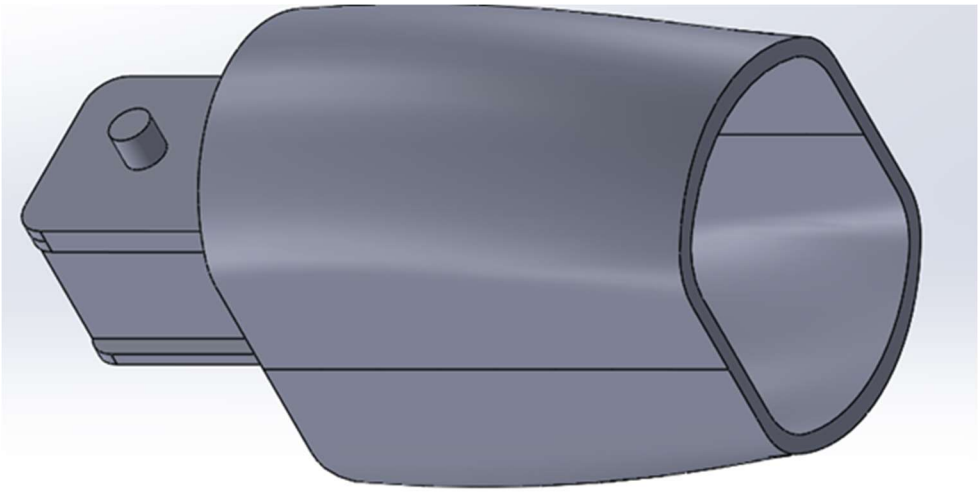


Figure 25: Secondary Forearm Prototype. The figure depicts the final shape of the forearm prior to claw attachments.

The similar profiles allowed the upper arm and forearm to be redesigned quickly to the

necessary size limits. The team was then able to print a completed arm assembly in order to test motor motion, and determine if any parts needed modification. As the first test worked well, a second arm and shoulder assembly were printed to test the six motors at a single time. Each arm sub-assembly consists of two motors for rotational and lateral shoulder movements and one for elbow movement. Figure 26, below depicts the arm design, prior to claw attachments.

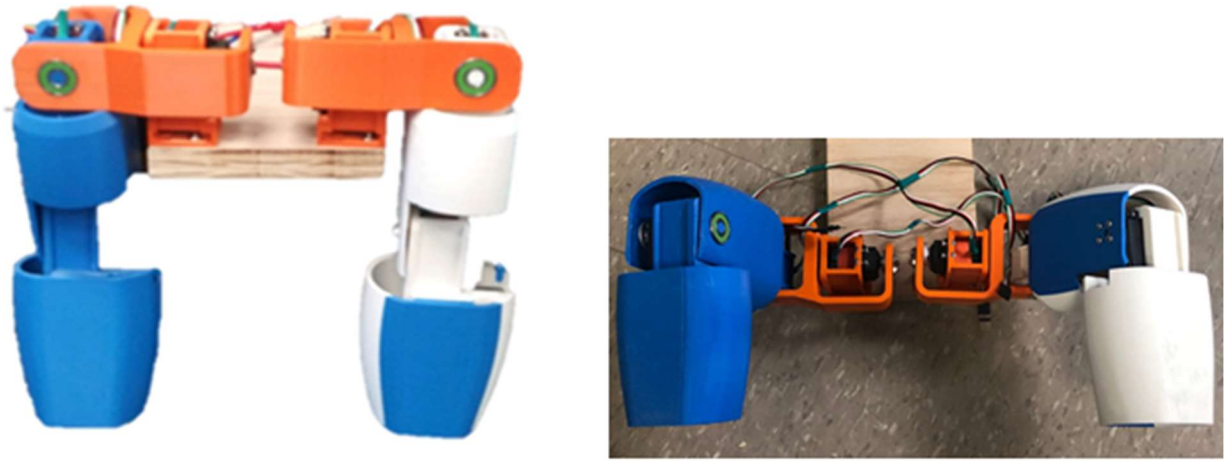


Figure 26: Final prototype of arm subassembly. Left image shows the front of the arm subassembly. The right image is a top view of the subassembly.

6.3. Wheels and Controller Subassembly

Comparatively to the time it took to see the arm subassembly prototype to completion, the wheels took a relatively short amount of time. After quickly laser cutting a temporary rectangular acrylic base to mount the mecanum wheels and portable breadboard to, code was written to test each of the movements of the wheels, starting with forwards and backwards, then progressing in complexity from there. A top view image of the acrylic base with the wheels attached and breadboard Velcro mounted is shown in figure 27. After each state of the wheels was found to work, the NES controller was wired to the portable breadboard to dictate the movements of the robot, as shown in Figure 28. Once the controller worked wired to the wheel's

breadboard, the RF module was added to a separate circuit to allow the user to control the robot without tethered to the robot. This required that an electrically erasable programmable read-only memory (EEPROM) chip be wired to the Propeller chip on the prototype wheel's circuit so that is would remember programs loaded after power was turned off then back on. The separate circuit for the RF module reading the NES controller was built on an MEB in the Mechatronic lab and is shown below in Figure 29. The hand-wired circuit for the wheels and the four h-bridge chips required to change the direction and power of the motors required many

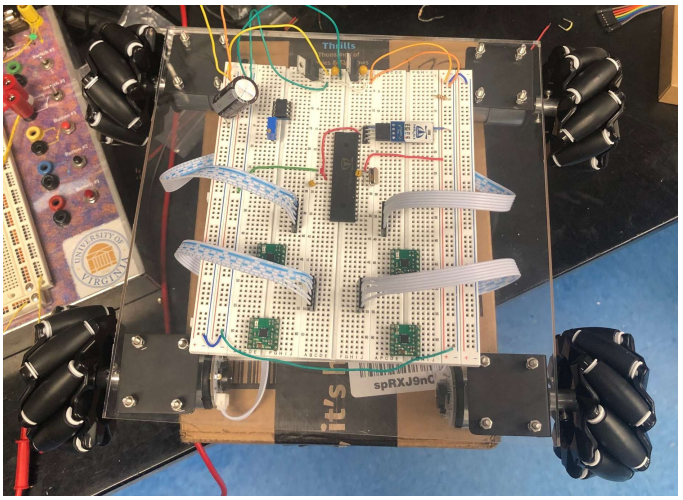
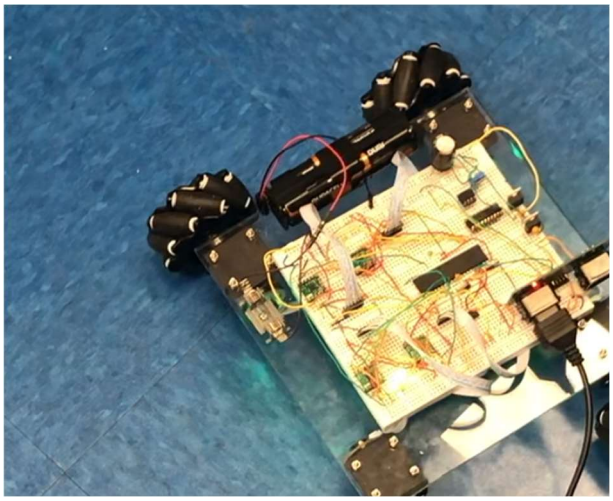


Figure 27 (Above): Velcro mounted bread board circuit.
Figure 28 (Below): NES controller attached to circuit.



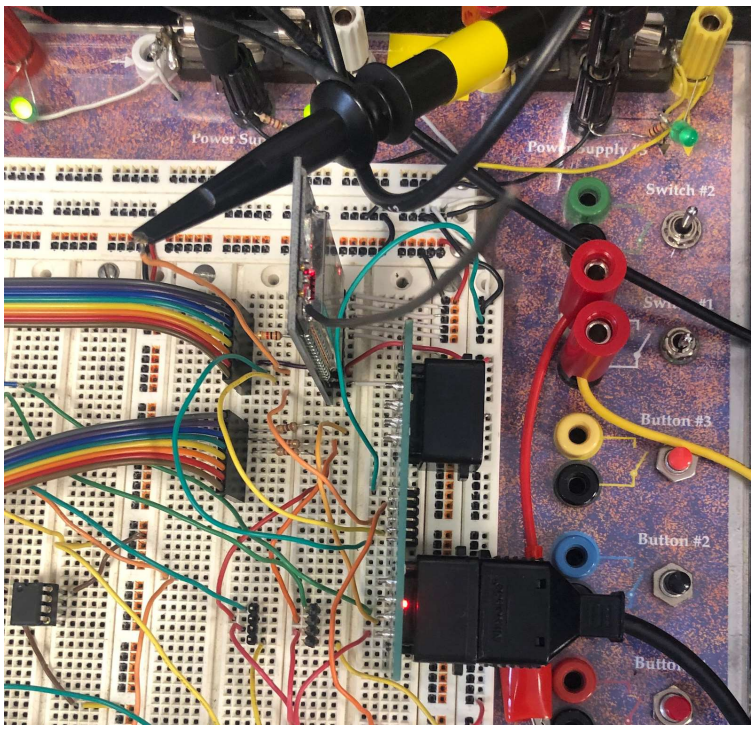


Figure 29: Bread board with NES controller and Transmitter card.

wires and got messy quickly. As soon as the wheels circuit was shown to work with the RF module and controller, Professor Garner assisted the team in making a perfboard for the wheel's circuit. This greatly neatened the circuit with the permanent soldered connections and can be seen in Figure 30 on the temporary acrylic base. The last change made to the wheels prototype was software rather than electronics or hardware based. As the team realized it was running out of available cogs nearing the final assembly of Rotundaur, Professor Garner assisted the team in authoring assembly code that code run all four of the wheel servos off of one cog, freeing up three cogs for the final code. This code can be seen in Appendix C1.

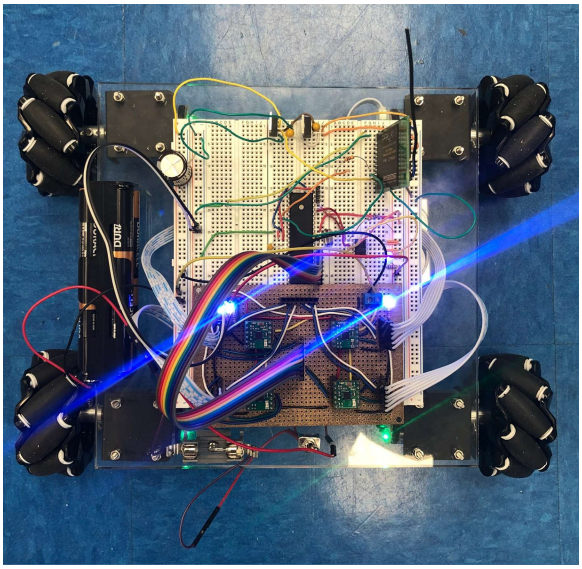


Figure 30: Wheel perfboard attached to acrylic base.

7. Manufacturing

One of the biggest benefits of choosing to 3D print much of the robot was that the team did not have to consider manufacturability in terms of using a general machine shop, since any geometry desired could be printed if it could be modeled in CAD. However, the team did have to consider assembly of parts, as small clearances between parts, particularly in the original joint prototypes necessitated redesigns so that the team could fit hardware where needed.

7.1. Dimension uPrint 3D Printer

Aside from CAD modeling in Solidworks, the critical manufacturing tool for Rotundaur were the Dimension uPrint 3D printers available to the team. These printers were used to fabricate essentially all parts of the arms, chest, and Rotunda using a solid density. STL files

were exported from Solidworks models and then imported into the CatalystEX 4.5 software to send to the 3D printers. The biggest concern for the team when printing parts was how they would be oriented during the print. The 3D printers used work through Fluid Deposition Modeling (FDM) where plastic filament is heated to a semi-liquid state then extruded in layers to form the desired part. Due to every part being printed in layers, the orientation of the part determines was direction it will be built up in - changing the properties and precision of the print in each direction. As a result, before printing prototypes, orientations were carefully discussed so that the parts would be printed as strong and precisely as possible. The team often had to sacrifice precision in one direction, usually choosing the one so that holes would be slightly not rounded and then drilling them out afterwards. While Professor Garner graciously did not charge the team for model or support material used in prototypes or the final parts, the team tried to carefully monitor printing amounts and times to not inconvenience other groups. In addition, the team tried to always double check parts were finalized and ready to be printed before starting a print.

7.2. PLS 6.75 Laser Cutter

The PLS6.75 Laser Cutter from Universal Systems was used to cut out the acrylic prototype base for wheel testing. This was chosen due to its ease of manufacture. As the name suggests, the laser cutter uses a laser beam within an enclosed box to cut, raster, or engrave the desired material. A rectangular prototype base was able to be designed in Solidworks, manufactured on the Laser Cutter out of ¼" thick acrylic, and assembled with the mecanum wheels in under an hour. Being able to have a functional base for the robot so quickly allowed for immediate testing and development of the wheel circuitry and code that was critical to the success of getting Rotundaur driving wirelessly. After being designed in Solidworks, a DXF file

was exported then imported into the CorelDraw graphic design program on the computer connected to the Laser Cutter. Afterwards, the PLS Software was used to finalize the file, oriented the laser to the proper location on the acrylic sheet, and confirm setting for the cut such as the material and thickness. The final base and other parts on the robot were chosen to not be machined from acrylic due to the ease with which it shatters. If one person had an aggressive encounter with the robot, a new base would need to be fabricated and all attached components rewired. As the laser cutter is optimally used for acrylic be decided to use the Waterjet to cut the final base out of Aluminum.

7.3. OMAX Maxiem 1515 Waterjet Cutter

The final base for Rotundaur was manufactured on the Maxiem 1515 Waterjet Cutter in the Lacy Hall Laboratory and Workshop available for use by students in the School of Engineering and Applied Science at UVA with the help of Sebring Smith. The waterjet is remarkably similar to the laser cutter in that both cut out 2D profiles from sheets of material using an imported CXF file, however, the waterjet is ideal for cutting through metals such as Aluminum. Instead of using a laser beam to cut the material, the waterjet uses a thin stream of garnet propelled forcefully at the sheet next to a beam of pressurized water in order to cut the desired material. As the bed of the waterjet is exposed, additional safety measures such as eye protection and hearing protection were undertaken. In addition, like the laser cutter, shapes that could typically not be machined traditionally easily such as partial circles can be cut out with ease on the waterjet. Holes and slots can also be cut out simultaneously as the waterjet cuts the sheet to shape rather than needing to use a saw, drill, and mill in traditional machining.

For Rotundaur's base, the team decided to construct it out of a piece of 8"x8" ⅜" thick 6061 Aluminum sheet form McMaster Carr. As the team was trying to save on budget, the sheet

was ordered to be as close to the size needed for the base so that there would not be excess material as the sheets quickly jump in price and size to a minimum of 2 feet in length. However, this actually made the machining process on the waterjet more dangerous, as clamping the piece down to cut out so it did not drastically move or bow upwards was difficult. The DXF file used is shown to the right in Figure 31.

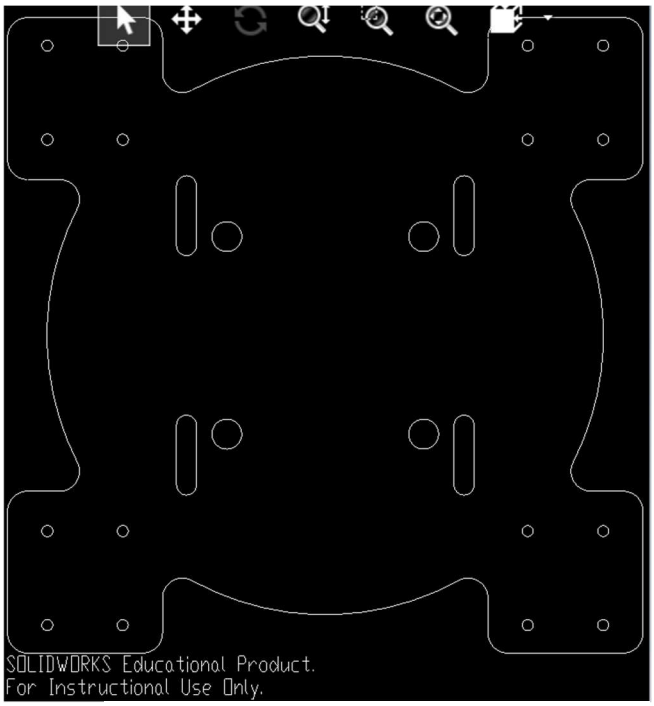


Figure 31: DXF file for water cutting aluminum base.

To manufacture the base, the DXF file was brought onto the waterjet's computer in the IntelliMAX Make software. The waterjet cut route was carefully selected with the assistance of Mr. Smith so that the error from the movement of the part when the jet initially cuts would be minimized. Afterwards, the machine was homed in the x,y, and z directions and then began the cut. Care had to be taken when refilling the garnet while the machine was in operation so it did not get even the smallest amount of moisture on it which can clog the nozzle and cause abrasive cuts or worse.

8. Final Assembly

In large disagreement with the original Gantt, final assembly was unfortunately largely conducted the week the project was due. The two biggest hindrances to beginning final assembly earlier were the printing times for parts, such as the forearms, torso, rotunda, and rotunda aesthetics, and not having the second perf board with the arms circuitry. Because colors of printing material mattered to the project and only a certain amount of blue and orange material

were available, the team had to be fully confident reprints would not be needed. After reviewing the model with Professor Garner and getting the okay to start printing, a printing schedule was constructed a week and a half before the project so a team member would always be available to remove a part and start a new one as soon as a print was completed. The most critical parts, the forearms and torso, were to be printed first in blue, followed by the Rotunda in orange and then rotunda aesthetics last in white. The print times of these parts were all large, between 15 and 32 hours. As there were six necessary prints to complete this, six full days were needed, bringing the team right up to the deadline. Final parts, such as the arms and claws, were assembled as soon as they were done printing so that final testing with the circuits could be done while less critical parts such as the torso were still printing. While this was happening, part of the team transferred the wheels and wheel perf board over to the final aluminum base plate, added fuses for all of the servo motors, and confirmed the wheels still worked with the NES controller commands, which it did. The two halves of the Rotunda with the aesthetics, the triangle piece and stairs, were constructed off of the base and one of them connected to the torso to speed up final assembly onto the base once the final perf board was ready to be put on after testing. Figure 32 below depicts the completely assembled exterior of Rotundaur.

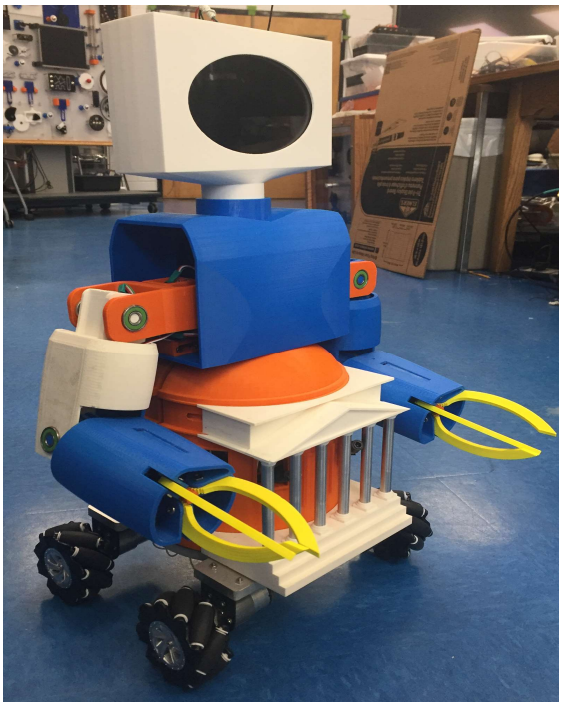


Figure 32: Completed Rotundaur. A completed Rotundaur robot with aesthetics attached.

The last component necessary for final assembly, the second perf board, containing the propeller chip, and hardware for the arms and claws, was received the night before the project was due. This severely limited the team's time in testing its functionality. For ease of access, testing was done with the Rotunda and torso off initially. Due to the number of connections on the board, several mishaps were expected to be found during testing. Known parts of the circuit working, such as the wheels were tested using the final board initially, and errors on the board corrected as they were found, such as a nonfunctional crystal oscillator and a missing connection for 3.3V to the propeller chip. Testing proceeded as such, gradually adding on complexity in order to isolate which circuit component was dysfunctional. Once testing for the wheel circuit connections was complete on the new perf board, the battery, two perf board, and 3D printed holder for the perf boards were mounted to the aluminum base of the robot. Many issues had to be worked out in this phase, as larger wires had to be added for the +12V and +5V power supplies to the board, wires soldered to the fuses for the servo motors, and a custom fuse and connector pack added onto the battery by Professor Garner. Once this was completed, wheel testing with the controller and RF modules was then completed until the wheels were functional off of the controller's commands like on the prototype.

After the wheels were working, the torso, half of the rotunda with aesthetics, arms, and claws were mounted. This required feeding many wires with connectors throughout the robot to attach the servo motors in the arms and the RF module in the head to the centrally located perf boards in the Rotunda center. The rest of the assembly time was spent attempting to get the arms to function in puppet mode as they had previously when wired to the testing circuit on the MEB. Prior to the final presentation of the robot, the issue was isolated to the ADC chip and software in the final code. While the ADC and code to sample it were found to work on the MEB board

when hooked up to a motor, when put onto the perf board, the ADC constantly read 0. Due to a lack of time, further testing into correcting this had to be sacrificed right before the presentation so that the robot could be fully assembled aesthetically. As a result, the final assembled robot did not have full functionality as expected from the functional prototype wheel and arm subassemblies.

9. Challenges

As with any worthwhile engineering project, many challenges were encountered along the way. The biggest challenges of the project were related to planning, CADing, manufacturing, and testing during final assembly. With regards to planning, while efforts were made to follow the Gannt chart, the time needed to design and prototype the motor joint and conduct tests on components was severely underestimated. In addition, waiting until the final weeks of the project to construct a full CAD assembly hindered the team as final printing couldn't commence until interferences and full robot assembly and disassembly were discussed and finalized. The biggest manufacturing challenge encountered was the length of the prints in the 3D printer and the tolerances. With little room between the project deadline and time needed to finish the scheduled prints, sacrifice in the quality of some prints had to be made to reduce print times. The biggest challenge the team faced was definitely in the final day leading up to the project deadline. The team worked from 5pm until 11am to test and complete the final assembly, but inadequate testing time with the final perf boards ultimately caused the final assembly to be non-functional. While it is unfortunate that Rotundaur did not function properly in the final presentation, the team is confident that with a few more hours of lower stress testing, she could be made fully functional.

10. Successes

While at first glance, the challenges of the project are more apparent in the final product than the successes, many successes were achieved throughout the duration of the project. The biggest technical successes were the subassemblies functioning, including, the wheels controlled off of the wireless NES controller, the arms successfully repeating motions in puppet mode, and the claws succeeding in picking up a rubber ball controlled by the NES controller. Each of these feats was achieved from the unison of a full mechatronic design, requiring hours of CAD modeling, manufacturing, designing the circuitry, writing software, and testing. The biggest personal feats as individuals on the team vary, but all team members are proud of their individual contributions to the bigger project. Each team member was able to take ownership of a certain part of Rotundaur, and see it from Solidworks through printing to functioning with the electronics. In addition, the team was able to function well collaboratively despite varying backgrounds and expertise.

11. Future Work

The most striking future work project for Rotundaur is to see her to full functionality within the team's project scope; with a working puppet mode and wheels within the fully assembled robot. However, other projects, such as adding a custom controller that is completely wireless (instead of wired to an NES adaptor and second propeller chip) would be a great addition. Easy additions, such as finishing adding the LED strips could be completed quickly in the future. A great safety feature of these LEDs could be to have them flash green at the start of puppet mode, yellow when recording is about to end, and red when the user should stop moving the arms as they are about to begin moving to repeat the motions. Furthermore, completing the screen and head so that a face can be displayed and changed would provide Rotundaur a larger personality as initially intended. Additionally, an SD card could be added to be able to store

previous puppet modes and repeat them at a later time, and preprogrammed motions with audio/video capabilities could be added to best allow Rotundaur to serve as the tour-guide robot intended for the Mechanical Engineering Building.

12. Conclusion

While the final assembly of Rotundaur ultimately failed, the semester overall was quite a success. At some point, each of the primary goals of the team were accomplished: there was a working "puppet-mode", the mecanum wheels and claw sub assembly worked seamlessly with the RF module and NES controller, and aesthetically the robot looked incredible. Countless skills were learned by members of the group; from machine shop skills work using the water jet, saw, and lathe; to mechatronic skills debugging breadboards and code. As mentioned earlier, each member of the team was responsible for designing a part of Rotundaur in Solidworks, and creating multiple assemblies with those parts. This reinforced the goal of the semester; that the engineer is the real final project, not Rotundaur. Each member of the team became a better engineer throughout the semester, whether it be through collaboration, learning the successful techniques of each other, or through the patient teaching in areas where another teammate might not be as strong. Each individual not only gained valuable technical experience, but also the intangible soft skills necessary to be successful in the engineering industry. The group experienced the importance of time management, especially in the concluding days of the project. The group learned the significance of dividing a large task into smaller and smaller manageable tasks and the importance of individual accountability within the larger project. Lastly, the group was fortunate enough to pursue a project each member was passionate about for the last time in an environment where it is totally safe to fail for the sake of learning, for this, the team is truly thankful to Professor Garner.

Every exciting breakthrough and seemingly impossible roadblock reminded the team of why each individual committed to Mechanical Engineering. Since the inception of Rotundaur, the team spent countless hours programming, designing, and building a robot that the team could all look back fondly on. The highs were euphoric and the lows were agonizing, but in the end, although the final product was just short of dazzling, the journey along the way was well worth it.

References

Amazon.com: Moebius 4WD 80mm Mecanum Wheel Robot Car Chassis Kit with DC 12V Encoder Motor for Arduino Raspberry Pi DIY Project STEM Toy: Industrial & Scientific. (n.d.). Retrieved November 25, 2019, from https://www.amazon.com/Moebius-Mecanum-Chassis-Encoder-Raspberry/dp/B07VSW8VTB/ref=sr_1_20?keywords=mecanum+wheel+robot&qid=1571693204&sr=8-20

FIRST robotics competition game and season info. (n.d.). Retrieved October 18, 2019, from FIRST website: https://www.firstinspires.org/home

Garcia, E., Jimenez, M. A., Santos, P. G. D., & Armada, M. (2007). The evolution of robotics research. *IEEE Robotics Automation Magazine*, *14*(1), 90–103. https://doi.org/10.1109/MRA.2007.339608

Mataric, M. J., Koenigh, N., & Field-Seifer, D. (2007). Materials for enabling hands-on robotics and STEM education. Conference paper, *AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education.* Retrieved October 28, 2019, from http://robotics.usc.edu/publications/media/uploads/pubs/536.pdf

New NES Nintendo Controller. (n.d.-a). Retrieved November 25, 2019, from Lukie Games website: https://www.lukiegames.com/nes-nintendo-new-controller.html

Skelton, G., Pang, Q., Yin, J., Williams, B. J., & Zheng, W. (2010). Introducing engineering concepts to public school students and teachers: Peer-based learning through robotics summer camp. *Review of Higher Education and Self-Learning. 3*(7), 8.

Sony Aibo ERS-7 | Sony Aibo. (n.d.). Retrieved November 25, 2019, from http://www.sony-aibo.com/aibo-models/sony-aibo-ers-7/

Welcome to Erector by Meccano ® The original inventor brand! (n.d.). Retrieved November 25, 2019, from http://www.meccano.com/product/p10945/meccanoid-g15ks-personal-robot

Appendices

Appendix A

A1 - Gantt Chart Table

GANTT CHART

Rotundaur - Machine Design Capstone Project Emily D, Russell H, Alyssa R, Win S

* = an automatically calculated cell

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	Notes	TEAM MEMBER	PERCENT COMPLE
Concept						All	
Define robot purpose	9/11	9/30	0	20			100%
Prior art/inspiration	9/11	9/30	0	20			100%
Hand Sketches	9/11	9/30	0	20			100%
Parts Decisions						All	
Wheels	9/11	9/30	0	20			100%
Motors (Wheels)	9/23	10/2	12	10			100%
Motors (Arms)	9/23	10/2	12	10			100%
Body	9/11	10/2	0	22			100%
Misc.	9/11	11/7	0	58			100%
Battery	9/23	11/11	12	50			100%
Controller	9/11	10/14	0	34	Future custom		50%
Design (Solidworks CAD Model)							
Claws	11/10	12/1	60	22		Emily	0%
Arm System	10/2	10/23	21	22		Russell	100%
Body	10/2	10/23	21	22		Win	100%
Rotunda	10/2	11/20	21	50		Alyssa	100%
Circuit Diagrams	10/2	10/30	21	29		Emily	100%
Full Robot Assembly	10/2	11/1	21	31		All, Russell	100%
Coding							
Mecanum Drive (NES Controller)	10/7	11/1	26	26		Emily	100%
Preprogrammed Motions	11/1	11/8	51	8			0%
Mimic/Puppet	11/8	11/18	58	11		All	90%
Controller Arms	11/18	11/26	68	9		All	0%
Build							
Prototypes	10/9	10/25	28	17			100%
3D Print Body Parts	10/23	11/3	42	12			100%
Wheel System & Base	10/11	10/21	30	11			100%
Arms	10/18	11/1	37	15			100%
Controller	10/9	10/18	28	10	NES Controller		50%
Housekeeping						All	
Budget	9/30	12/12	19	74			100%
Checkpoint #1 Presentation	9/16	9/23	5	8			100%
Checkpoint #2 Presentation	9/25	10/2	14	8			100%
Checkpoint #3 Presentation	10/9	10/16	28	8			100%
Checkpoint #4 Presentation	10/21	10/28	40	8			100%
Checkpoint #5 Presentation	11/4	11/11	54	8			100%
Checkpoint #6 Presentation	11/13	11/20	63	8			100%
Final Presentation	11/24	12/12	74	19			100%
Final Technical Report	11/24	12/15	74	22			100%

Table A1 – Table for the Gantt Chart for the Rotundaur project.
Schedule for the semester long project set during the first week. While often not followed to a tee, the schedule helped the team track the project's progress and remember what still needed to be done.

Appendix A2 – Gantt Chart

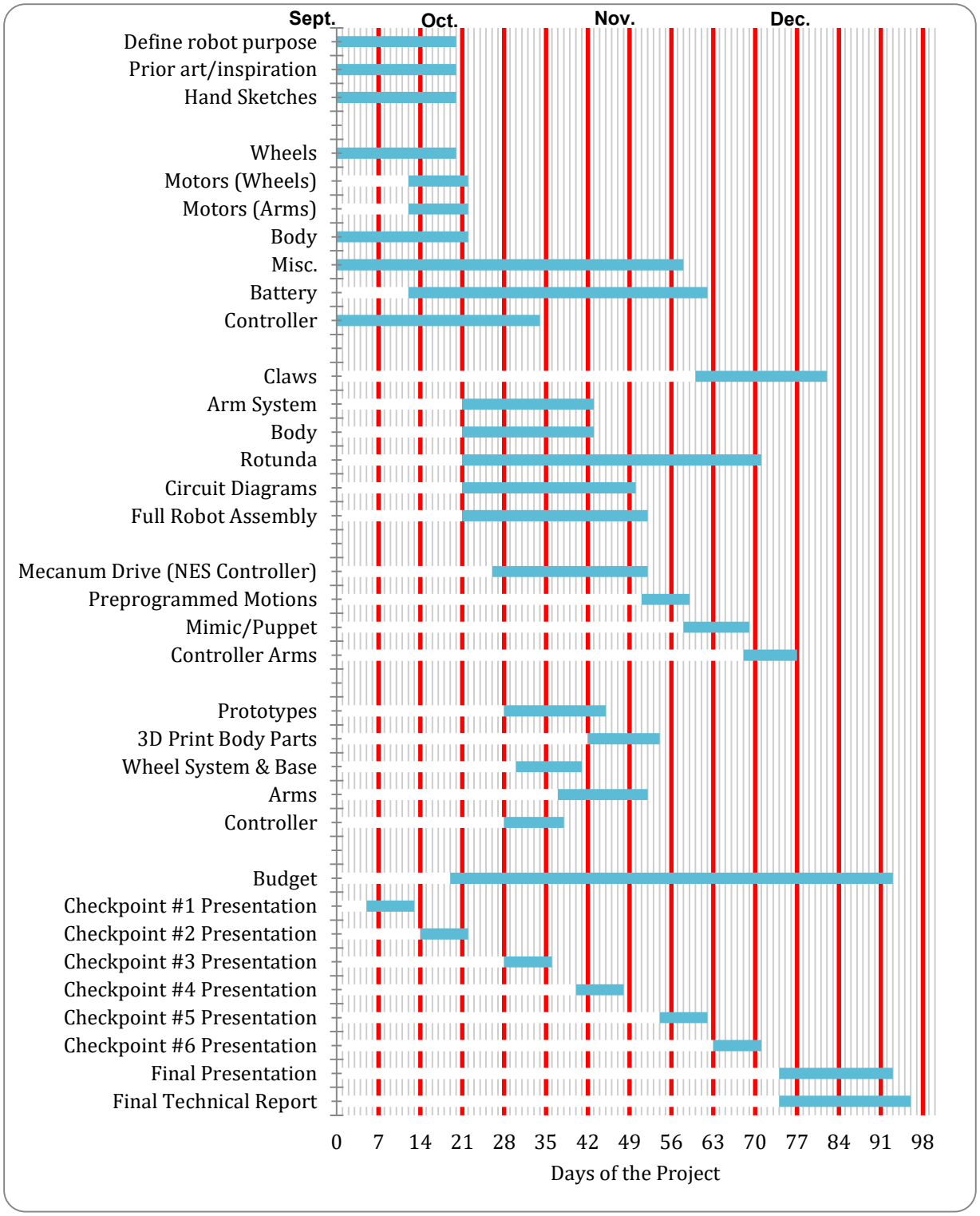


Figure A2 – Gantt Chart for the Rotundaur project. Line items correspond to Table A1 and months of the project are labeled along the top.

Appendix B

B1 – Budget and Parts List

Item	Retailer	Part #	Price	Qt.	Total
80 mm Macnum Wheels, Motors, & Base kit	Amazon	B07D56FVK5	$84.49	1	$ 84.49
10pcs. 25T Servo disc	Amazon	B07D56FVK5	$8.99	1	$ 8.99
270deg. Digital Servo Motors	Amazon	B07HNTKSZT	$15.99	8	$ 127.92
8" x 8", 3/8" thick Aluminum 6061 sheet for wheel base	McMaster	9246K21	$26.11	1	$ 26.11
3/8"-16 foot long threaded rods (attach torso to rotunda)	McMaster	98790A058	$0.81	8	$ 6.48
6-32 pan head philips steel screws (100 per pkg)	McMaster	90272A146	$2.19	1	$ 2.19
3/8"-16 hex nut steel grade 5	McMaster	95505A603	$5.66	1	$ 5.66
3/8" lock washers	McMaster	92147A031	$6.37	1	$ 6.37
6-32 lock washers	McMaster	92146A535	$1.64	1	$ 1.64
PQ12-R linear actuator, 6volt, 100:1 ratio	Actuonix	PQ-12R	$70.00	2	$ 140.00
THINGS WE DIDN'T PAY FOR WE SHOULD HAVE					
DRV8801 Single H-Bride Chip from Mechatronics Kit	Texas Instruments	DRV8801	$5.00	6	$30.00
TLC2543 12-bit Multiplexer ADC Chip	Texas Instuments	TLC2543	$12.00	1	$12.00
1A Fuse	-	-	$0.15	30	$4.50
Adafruit RB LED Neopixel Stick	Adafruit	1426	$5.95	5	$29.75
Serial RF Module	Parallax Inc.	-	$50.00	2	$100.00
NES Controller	-	-	$5.00	1	$5.00
MMG YTZ10S Z10S Rechargeable Lithium Ion Sealed Battery 12V	Amazon	MMG_YTZ10S_LI	$103.90	1	$103.90
Parallax 40-DIP Propeller Chip	Parallax Inc.	P8X32A-D40	$7.99	2	$15.98
Misc. Circuitry (Jumper Wires, Connectors, Resistors, etc.)	-	-	-	-	$10.00
Bearings	-	-	$10.00	6	$60.00
Misc. Hardware (Nuts, Bolts, Washers, etc.)	McMaster	-	$0.10	50	$5.00
Stratasys ABSPlus Model XL Material (30in^3)	-	-	$200	6	$1,200.00
Stratasys ABSPlus Support XL Material (30in^3)	-	-	$200	4	$800.00
Professor Garner's Time and Energy	UVA	MAE4610/20	-	-	Priceless

TOTAL BUGET	750
Remaining Budget	$ 340.15
Including Not-Charged	$(2,035.98)

Table B1 – Budget and Parts List for the Rotundaur project. The top items in the table are material paid for out of the team's budget of $750, highlighted in yellow in the top right. Items highlighted in green were the most expensive. The bottom table lists parts not charged to the team but used on the robot, with items in red being the most expensive. $340.15 was left of the budget at the end, shown in the top right in green, including only the charged items. The project is technically $2035 over budget if estimates for not charged parts are included.

Appendix C - Circuitry Layout and Software

C1.1 - Wheels Circuitry Layout

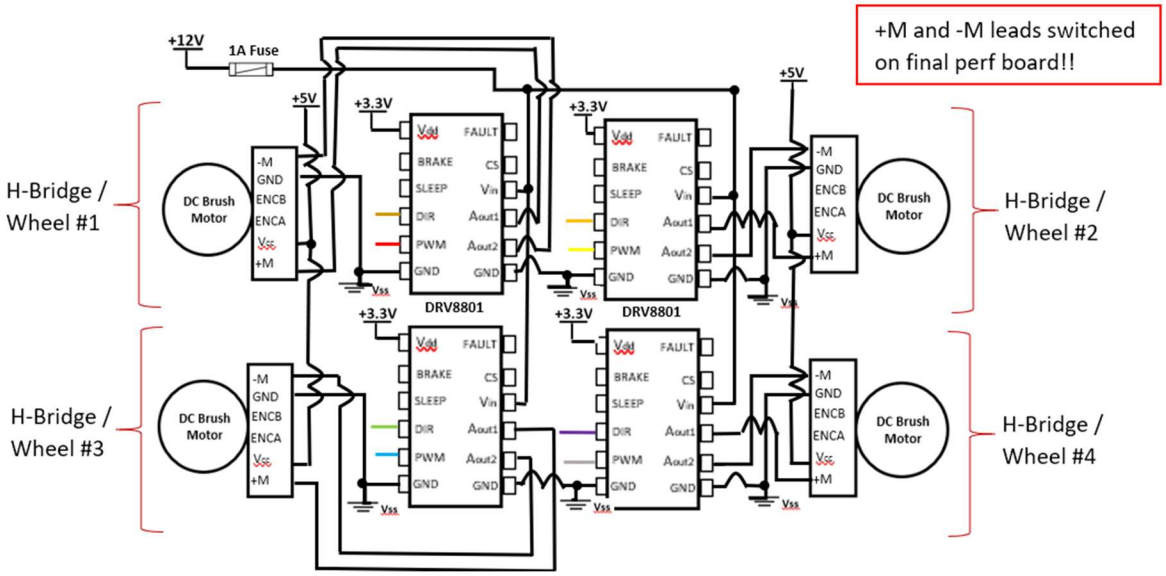


Figure C1 – Circuit and Wiring Diagram for the Mecanum Wheels and H-Bridges.
The final perfboard used on Rotundaur switched the +M and -M leads, which changes whether 1 or 0 spins the wheels CW or CCW. This is accounted for in the software below.

C1.2 – Wheels Software

```
CON
  _xinfreq = 6_250_000
  _clkmode = xtal1 + pll16x

  RX_pin=24             'RF module pin that recieves data about the NES controller's state
  TX_pin=5              'pin is not used so definition is arbitrary, but needed for the fds.Start() command

  Dir1=20               'Directional control; pin 1=CCW 0=CW when looking @ the motor
  Dir2=21
  Dir3=22
  Dir4=23


  PWM1=16               'PWM pin controls the signal sent out by the H-Bridge chip to control the motor's speed
  PWM2=17
  PWM3=18
  PWM4=19

  LinServoL=11
  LinServoR=12

VAR
  byte databyte                                 'variable for the NES controller's state recieved
  long  speed1,speed2,speed3,speed4             'speed = DutyCycle = hgih time of PWM signal (0-100%)
  long positionL, positionR                     'linear servos positions for the claws
        '200_000 = 2ms pulse = fully retract
        '100_000 = 1ms pulse = fully extend

PUB Main
  fds.Start(RX_pin,TX_pin,0,9600)          'Start(rxPin, txPin, mode, baudrate) -- This is running on Cog 1
  s.Start                                  'Starts on Cog2

  dira[RX_pin]~
```

```
dira[TX_pin]~~

'Taken from QuadPWM COde from Garner
_speed1:=@speed1
_speed2:=@speed2
_speed3:=@speed3
_speed4:=@speed4

speed1:=35
speed2:=35
speed3:=35
speed4:=35

coginit(3,@QuadPWM,0)                    ' Start QuadPWM assembly code on Cog 3

dira[PWM1..Dir4]:=%1111_1111          'Set all the H-Bridge PWM and Dir pins to outputs

positionR:=200_000
positionL:=200_000

repeat
  databyte:=fds.rx

  'Read which buttons on the NES controller are being pressed (this data is refreshed during "case" and stored as Databyte
  'In this case, %1111_1101 indicates that the LEFT button is being pressed (since there is a zero in that bit)
  'NES controllers use a bit for each button and inverted logic in this order:
  'MSB=A_button__B_button__Select__Start/____Up__Down__Left__Right=LSB
  speed1~
  speed2~
  speed3~
  speed4~

  case Databyte              'Check state of the NES controller

    %1111_1111 :          'If no buttons pressed, stop moving
      speed1~
      speed2~
      speed3~
      speed4~

    %1001_1111:         'SELECT button & B pressed (Left Claw open)
      positionL:=(positionL-500)          #>100_000
      s.Set(LinServoL, positionL/100)               'Use Servo32 object to run LinServos
      waitcnt(clkfreq/100+cnt)

    %0101_1111:         'SELECT button & A pressed (Left Claw close)
      positionL:=(positionL+500)          <#200_000
      s.Set(LinServoL, positionL/100)
      waitcnt(clkfreq/100+cnt)


    %1010_1111:         'START button & B pressed (Right Claw open)
      positionR:=(positionR-500)          #>100_000
      s.Set(LinServoR, positionR/100)
      waitcnt(clkfreq/100+cnt)
```

```
x0110_1111:         'START button & A pressed RLight Claw close)
  positionR:=(positionR+500)          <#200_000
  s.Set(LinServoR, positionR/100)
  waitcnt(clkfreq/100+cnt)


x1111_0111 :          'UP button pressed  (forwards)
  outa[Dir1..Dir4]:=x1010
  {outa[Dir2]:=0
  outa[Dir3]:=1
  outa[Dir4]:=0}

  speed1:=35
  speed2:=35
  speed3:=35
  speed4:=35

x1111_1011 :          'DOWN button pressed  (backwards)
  outa[Dir1..Dir4]:=x0101
  {outa[Dir2]:=1
  outa[Dir3]:=0
  outa[Dir4]:=1}

  speed1:=35
  speed2:=35
  speed3:=35
  speed4:=35

x1111_1101 :        'LEFT button is being pressed  (strafe left)
  outa[Dir1..Dir4]:=x0011
  {outa[Dir2]:=0
  outa[Dir3]:=1
  outa[Dir4]:=1}

  speed1:=25
  speed2:=25
  speed3:=25
  speed4:=25

x1111_1110 :         'RIGHT button is being pressed   (strafe right)
  outa[Dir1..Dir4]:=x1100
  {outa[Dir2]:=1
  outa[Dir3]:=0
  outa[Dir4]:=0}

  speed1:=35
  speed2:=35
  speed3:=35
  speed4:=35

x1111_0110 :        'UP and RIGHT pressed (diagonal NE)

  outa[Dir1..Dir4]:=x1000      'CCW for 1 and CW for 4; 2 and 3 do not matter

  speed1:=35
  speed2:=0             'No power to wheels 2 and 3
  speed3:=0
  speed4:=35

x1111_0101 :          'UP and LEFT pressed (diagonal NW)
  outa[Dir1..Dir4]:=x0010              '2 is CW and 3 is CCW; 1 and 4 do not matter

  speed1:=0
  speed2:=35
  speed3:=35
  speed4:=0

x1111_1010 :         'DOWN and RIGHT pressed (diagonal SE)
  outa[Dir1]:=x0100              '2 is CCW and 3 is CW ;1 and 4 do not matter

  speed1:=0
  speed2:=35
  speed3:=35
  speed4:=0

x1111_1001 :        'DOWN and LEFT pressed (diagonal SW)
  outa[Dir1..Dir4]:=x0001              '1 is CW and 4 is CCW; 2 and 3 do not matter

  speed1:=35
  speed2:=0
  speed3:=0
  speed4:=35

x0111_1111 :          'A button is being pressed (Rotate LEFT/CCW)
  outa[Dir1..Dir4]:=x1111   'All CW

  speed1:=35
  speed2:=35
  speed3:=35
  speed4:=35
```

```
    %1011_1111 :          'B button is being pressed (Rotate RIGHT/CW)
      outa[Dir1..Dir4]:=%0000          'All CCW

      speed1:=35
      speed2:=35
      speed3:=35
      speed4:=35


DAT
            org
QuadPWM     mov dira,PinMask
Loop1       rdlong duty1,_speed1          'Update duty cycle values from central Hub RAM
            add duty1,#1
            rdlong duty2,_speed2
            add duty2,#1
            rdlong duty3,_speed3
            add duty3,#1
            rdlong duty4,_speed4
            add duty4,#1

            mov LoopCounter,#101
Loop2       sub LoopCounter,#1 wz         'Subtract 1 from LoopCounter, if LoopCounter==0, set Z flag
      if_z  jmp #Loop1                    'If Loop2 has run 10,000 times, return to Loop1
            cmp duty1,#0 wz
      if_nz sub duty1,#1 wz                   'Z flag is set if duty1==0
      if_z  mov state1,#%0000                  'If countdown has reached zero (Z flag is set), pin should be turned off
      if_nz mov state1,#%0001                  'If countdown has not reached zero (Z flag is not set), pin should stay on
            cmp duty2,#0 wz
      if_nz sub duty2,#1 wz
      if_z  mov state2,#%0000
            cmp duty3,#0 wz
      if_nz sub duty3,#1 wz
      if_z  mov state3,#%0000
      if_nz mov state3,#%0100
            cmp duty4,#0 wz
      if_nz sub duty4,#1 wz
      if_z  mov state4,#%0000
      if_nz mov state4,#%1000
            add state1,state2               'Combine all four states
            add state1,state3
            add state1,state4
            shl state1,#16
            mov outa,state1                 'Set all four pins to the correct on/off states
            jmp #Loop2

PinMask     long    %1111 << 16           'Set pins 16-19 to be outputs
outputs     long    %0000
_speed1     long    0       'Place to store central Hub RAM address of "speed1" variable on this Cog's RAM
_speed2     long    0
_speed3     long    0
_speed4     long    0
timer       res
LoopCounter res
duty1       res
duty2       res
duty3       res
duty4       res
state1      res
state2      res
state3      res
state4      res
StartTime   res
            fit
```

Figure C1.2 – Software to control Mecanum Wheels with RF Module.
Case loop continuously updates the buttons being pushed on the NES controller to move the robot and open/close the claws. Assembly code allows a PWM signal to be sent to the four mecanum wheels to control their speed using only one cog on the Propeller chip.

Appendix C2 – Controller Circuitry Layout and Software

C2.1 – Controller Circuit Layout

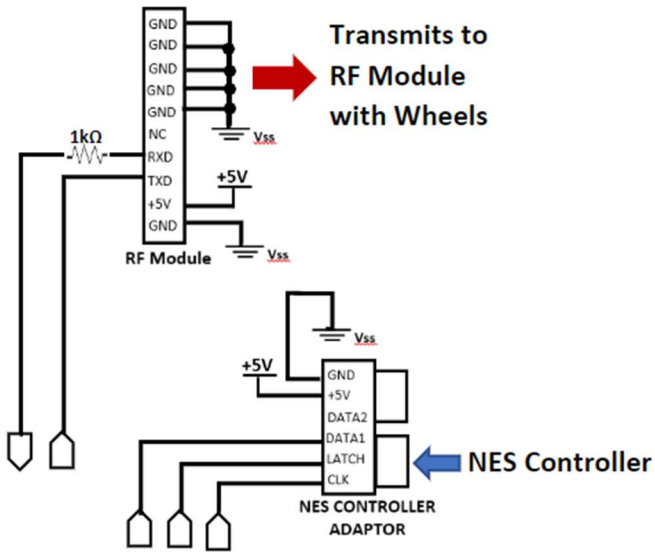


Figure C2.1 – Circuit and wiring diagram for the separate off-robot controller.
An NES controller plugs into the adaptor in the circuit which interfaces with the propeller chip and a second RF module set as a transceiver. The module sends a databyte of information to the receiver connected to the propeller chip on Rotundaur, dictating how the wheels and claws should move.

C2.2 – Controller Software

```
CON
  _xinfreq = 6_250_000
  _clkmode = xtal1 + pll16x

  RX_pin=4
  TX_pin=3
  identifier=77

'NES Controller interface pins
  clk=0
  latch=1
  data1=2

OBJ
  fds : "FullDuplexSerial"

VAR
  byte C1buttons

PUB Main

  fds.Start(RX_pin,TX_pin,0,9600)

  dira[clk..latch]~~              'Sets I/O directions of NES Controllers' clock and latch interface pins to be outputs
  dira[RX_pin]~           'input
  dira[TX_pin]~~          'output


  repeat
    UpdateAll
    fds.tx(C1buttons)           'Transmit C1 buttons databyte to the reciever on the wheels
PUB UpdateAll                   'Read the NES controller's button states
  C1buttons~
  outa[latch]~~
  C1buttons:=ina[data1]
  outa[latch]~
  repeat 7
    outa[clk]~~
    C1buttons:=C1buttons<<1+ina[data1]
    outa[clk]~
  outa[clk]~~
  outa[clk]~
```

Figure C2.2 – Controller software to read the state of the NES buttons and transmit it to the receiver on the robot.

The 'UpdateAll' method refreshes the state of the buttons continuously in the repeat loop in the Main method, which then sends it to the matching RF module on Rotundaur.

Appendix C3 – Full Robot Circuitry Layout and Software

C3.1 – Full Robot Circuitry Layout

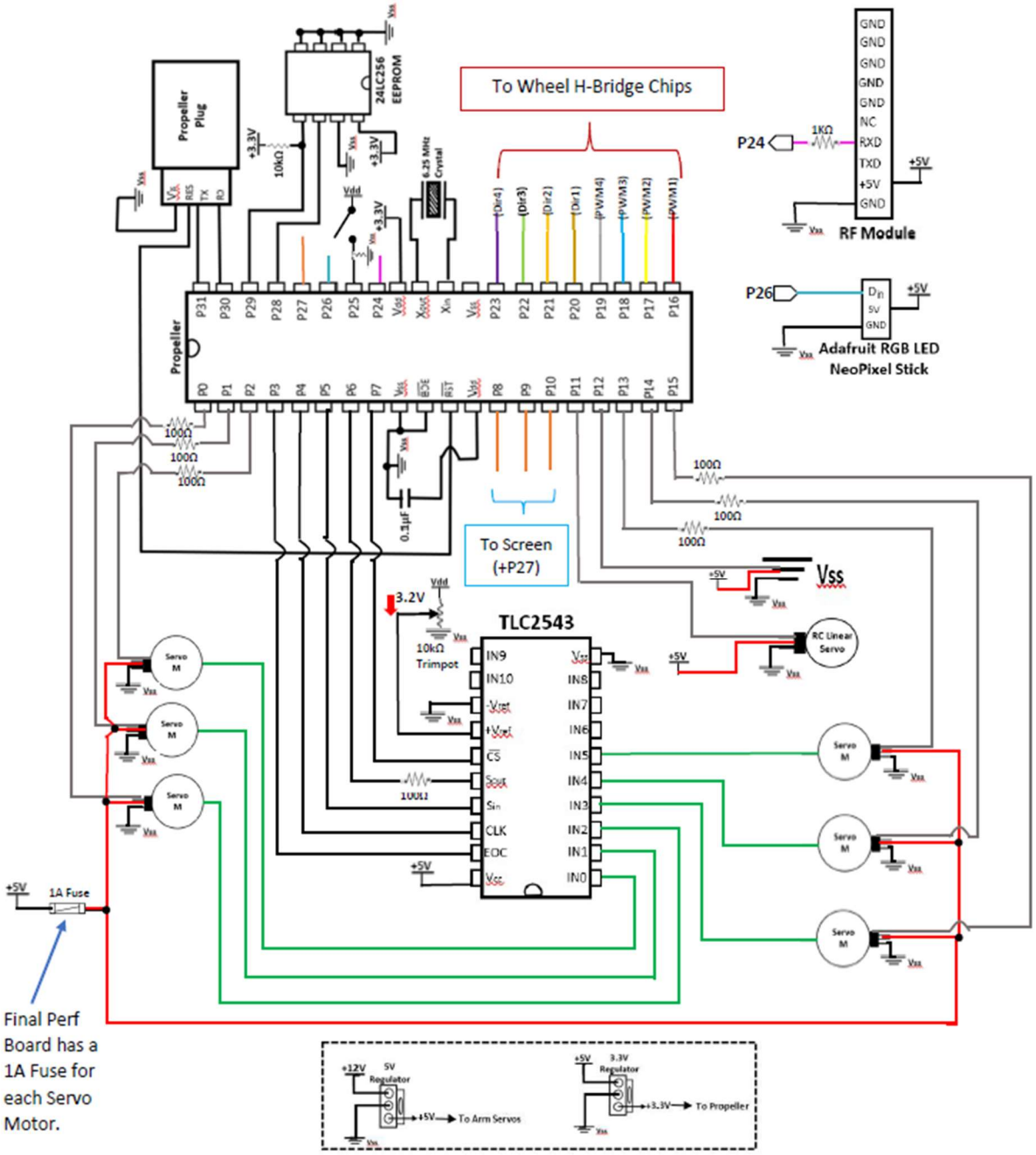


Figure C3.1 – Circuit and wiring diagram for the Propeller chip on the full Rotundaur robot. All pins are currently used, except for P8-10 and P27 which can be wired to a screen as outputs, and P26 for the LEDs in the future.

C3.2 – Full Robot Software

```
CON
  _xinfreq = 6_250_000
  _clkmode = xtal1 + pll16x

  'Pins for the RF module on the robot to communicate with the remote
  RX_pin=24
  TX_pin=25     'not connected

  'Pins for the Wheel Motor's H-Brigde Chips
  Dir1=20
  Dir2=21
  Dir3=22
  Dir4=23             'Directional control pin 1=CCW 0=CW when looking @ the motor

  PWM1=16             'PWM pins form H-bridge to Propeller chip
  PWM2=17
  PWM3=18
  PWM4=19

  'Pins for the Servos on the Arms
  Servo0=2       'Servo in ADC0 is Pin2
  Servo1=1
  Servo2=0
  Servo3=15
  Servo4=14
  Servo5=13

  LinServoL=11
  LinServoR=12

  Puppet_Trigger=27             'When this ionput pin goes high (3.3V) start puppet mode

  'Pins for the ADC Multiplexer Chip
  eoc=3
  clk=4
  Sin=5
  Sout=6
  cs=7

  'Pin for the LEDs Din
  TotalLEDs=24   '<---set the maximum number of LEDs you wish to control (5*8 = 40)
  LED_Data=26

  screen1=8         '3 different images to display on the head's screen
  screen2=9
  screen3=10

OBJ
  fds : "FullDuplexSerial"
  PWM : "QuadPWM"
  pst : "PST_Driver"
  s : "servo32v7"

VAR
  byte databyte
  long speed1,speed2,speed3,speed4                      'speed = DutyCycle = hgih time of PWM signal (0-100%)
  long positionL, positionR, servoPos0
       '200_000 = 2ms pulse = fully retract
       '100_000 = 1ms pulse = fully extend
  word ADCdata0, ADCdata1, ADCdata2, ADCdata3, ADCdata4, ADCdata5               'Variable to store a word from each ADC channel we sample
  word position_data0[1000], position_data1[1000], position_data2[1000], position_data3[1000], position_data4[1000], position_data5[1000]
  long Stack6[100]
```

```
PUB Main

  fds.Start(RX_pin,TX_pin,0,9600)          'Start(rxPin, txPin, mode, baudrate) -- This is running on Cog 1
  s.Start                                  'Starts on Cog2
  s.Ramp                                   'Cog 3
  coginit(4,PuppetMode,@Stack6)

  dira[RX_pin]~
  dira[TX_pin]~~

  dira[puppet_trigger]~

    speed1~
    speed2~
    speed3~
    speed4~

    case Databyte                                                 'Check state of player 1's controller buttons

      %1111_1111 :          'If no buttons pressed, stop moving
        speed1~
        speed2~
        speed3~
        speed4~

      %1001_1111:         'SELECT button & B pressed (Left Claw open)
        positionL:=(positionL-500)           #>120_000
        s.Set(LinServoL, positionL/100)                'Use Servo32 object to run LinServos
        waitcnt(clkfreq/100+cnt)

      %0101_1111:         'SELECT button & A pressed (Left Claw close)
        positionL:=(positionL+500)           <#200_000
        s.Set(LinServoL, positionL/100)
        waitcnt(clkfreq/100+cnt)


      %1010_1111:         'START button & B pressed (Right Claw open)
        positionR:=(positionR-500)           #>120_000
        s.Set(LinServoR, positionR/100)
        waitcnt(clkfreq/100+cnt)


      %0110_1111:         'START button & A pressed RLight Claw close)
        positionR:=(positionR+500)           <#200_000
        s.Set(LinServoR, positionR/100)
        waitcnt(clkfreq/100+cnt)


      %1111_0111 :          'UP button pressed (forwards)
        outa[Dir1..Dir4]:=%1010

        speed1:=35
        speed2:=35
        speed3:=35
        speed4:=35

      %1111_1011 :          'DOWN button pressed (backwards)
        outa[Dir1..Dir4]:=%0101


        speed1:=35
        speed2:=35
        speed3:=35
        speed4:=35


      %1111_1101 :          'LEFT button is being pressed  (strafe left)
        outa[Dir1..Dir4]:=%0011

        speed1:=25
        speed2:=25
        speed3:=25
        speed4:=25

      %1111_1110 :           'RIGHT button is being pressed   (strafe right)
        outa[Dir1..Dir4]:=%1100

        speed1:=35
        speed2:=35
        speed3:=35
        speed4:=35

      %1111_0110 :          'UP and RIGHT pressed (diagonal NE)

        outa[Dir1..Dir4]:=%1000       'CCW for 1 and CW for 4; 2 and 3 do not matter

        speed1:=35
        speed2:=0              'No power to wheels 2 and 3
        speed3:=0
        speed4:=35
```

```
    X1111_0101 :          'UP and LEFT pressed (diagonal NW)
      outa[Dir1..Dir4]:=%0010               '2 is CW and 3 is CCW; 1 and 4 do not matter

      speed1:=0
      speed2:=35
      speed3:=35
      speed4:=0

    X1111_1010 :         'DOWN and RIGHT pressed (diagonal SE)
      outa[Dir1]:=%0100                 '2 is CCW and 3 is CW ;1 and 4 do not matter

      speed1:=0
      speed2:=35
      speed3:=35
      speed4:=0

    X1111_1001 :        'DOWN and LEFT pressed (diagonal SW)
      outa[Dir1..Dir4]:=%0001               '1 is CW and 4 is CCW; 2 and 3 do not matter

      speed1:=35
      speed2:=0
      speed3:=0
      speed4:=35

    X0111_1111 :          'A button is being pressed (Rotate LEFT/CCW)
      outa[Dir1..Dir4]:=%1111   'All CW

      speed1:=35
      speed2:=35
      speed3:=35
      speed4:=35

    X1011_1111 :           'B button is being pressed (Rotate RIGHT/CW)
      outa[Dir1..Dir4]:=%0000           'All CCW

      speed1:=35
      speed2:=35
      speed3:=35
      speed4:=35

PUB PuppetMode    | i, endtime, samples

  dira[puppet_trigger]~          'set puppet mode trigger pin to be an input
  outa[cs]~~                     'Start the ADC's chip select pin High

  ADC(0)                         'Set the initial channel to read the ADC on
  samples:= 749                  'Set number of smaples you want to get of position data  (750*.02 = 15s)
  'time:=  'sampling time of ADC loop for all 6 servos    '20us = 20_000 ms

  repeat
    if ina[puppet_trigger]==1

      repeat i from 0 to samples
        endTime:=cnt+(2_000_000)       ' 20ms adjusted to Servo32 code's period

        ADCdata0:=ADC(1)               'Set ADC multiplexer to perform NEXT conversion on Channel 1 (Motor1's output)
        position_data0[i]:=ADCdata0
        'pin 8
        ADCdata1:=ADC(2)               'Set ADC multiplexer to perform next conversion on Channel 2 (Motor2's)
        position_data1[i]:=ADCdata1
        'pin 10
        ADCdata2:=ADC(3)               'Set ADC multiplexer to perform next conversion on Channel 3 (Motor3's)
        position_data2[i]:=ADCdata2
        'pin12
        ADCdata3:=ADC(4)               'Set ADC multiplexer to perform NEXT conversion on Channel 4 (Motor4's)
        position_data3[i]:=ADCdata3
        'pin 9
        ADCdata4:=ADC(5)               'Set ADC multiplexer to perform next conversion on Channel 6 (Motor5's)
        position_data4[i]:=ADCdata4
        'pin 11
        ADCdata5:=ADC(0)               'Set ADC multiplexer to perform next conversion on Channel 0 (Back to Motor0's)
        position_data5[i]:=ADCdata5
        waitcnt(endTime)              'Wait for exactly 20ms every sample (adjusted period for Servo32)

        repeat i from 0 to samples
          s.Set(Servo0,(position_data0[i]*49+50_000)/100)          'Set(MotorPin#,PulseWidth)
          s.SetRamp(Servo0,(position_data0[i]*49+50_000)/100,20)       'SetRamp(MotorPin#,PulseWidth,Delay)
                                    'scale back to between 50_000 and 250_000, then convert to servo32 range by div 100
                                    '20 ms delay so motors can move
          s.Set(Servo1,(position_data1[i]*49+50_000)/100)
          s.SetRamp(Servo1,(position_data1[i]*49+50_000)/100,20)
          s.Set(Servo2,(position_data2[i]*49+50_000)/100)
          s.SetRamp(Servo2,((position_data2[i]*49+50_000)/100),20)
          s.Set(Servo3,(position_data3[i]*49+50_000)/100)
          s.SetRamp(Servo3,(position_data3[i]*49+50_000)/100,20)
          s.Set(Servo4,(position_data4[i]*49+50_000)/100)
          s.SetRamp(Servo4,(position_data4[i]*49+50_000)/100,20)
          s.Set(Servo5,(position_data5[i]*49+50_000)/100)
          s.SetRamp(Servo5,(position_data5[i]*49+50_000)/100,20)

        repeat until ina[puppet_trigger]==0                 'wait until pin goes high to go back into repeat loop
```

```
PUB ADC(NextChannel) : ADCbits | ADCdataByte, z
  dira[eoc..cs]:=%01101          'Set directions of pins connected to chip as inputs and outputs
  ADCbits:=0
  outa[clk]~                     'Set the clock pin low
  outa[cs]~                      'Set chip select pin Low (ready to read)  - inverted logic
  ADCdataByte:=NextChannel<<8+%0000<<4                    'Set channgel to do next conversion on & ADC mode (0000)
  waitpeq(|<eoc,|<eoc,0)         'Wait for eoc pin to indicate end of previous conversion
  repeat z from 11 to 0
    ADCbits:=(ADCbits<<1)+ina[Sin]                    'Shift previous bits left & add pin's current state
    outa[Sout]:=ADCdataByte>>z
    outa[clk]~                   'Set clock pin Low
    outa[clk]~~                  'Set clock pin High (Data is transferred across the SPI)
    outa[clk]~                   'Set clock pin low
  outa[cs]~~                     'Set chip select pin high (go to sleep)

DAT
              org
QuadPWM       mov dira,PinMask
Loop1         rdlong duty1,_speed1          'Update duty cycle values from central Hub RAM
              add duty1,#1
              rdlong duty2,_speed2
              add duty2,#1
              rdlong duty3,_speed3
              add duty3,#1
              rdlong duty4,_speed4
              add duty4,#1

              mov LoopCounter,#101
Loop2         sub LoopCounter,#1 wz        'Subtract 1 from LoopCounter, if LoopCounter==0, set Z flag
       if_z   jmp #Loop1                   'If Loop2 has run 10,000 times, return to Loop1
              cmp duty1,#0 wz
       if_nz  sub duty1,#1 wz                    'Z flag is set if duty1==0
       if_z   mov state1,#%0000                  'If countdown has reached zero (Z flag is set), pin should be turned off
       if_nz  mov state1,#%0001                  'If countdown has not reached zero (Z flag is not set), pin should stay on
              cmp duty2,#0 wz
       if_nz  sub duty2,#1 wz
       if_z   mov state2,#%0000
       if_nz  mov state2,#%0010
              cmp duty3,#0 wz
       if_nz  sub duty3,#1 wz
       if_z   mov state3,#%0000
       if_nz  mov state3,#%0100
              cmp duty4,#0 wz
       if_nz  sub duty4,#1 wz
       if_z   mov state4,#%0000
       if_nz  mov state4,#%1000
              add state1,state2              'Combine all four states
              add state1,state3
              add state1,state4
              shl state1,#16
              mov outa,state1                'Set all four pins to the correct on/off states
              jmp #Loop2

PinMask       long      %1111 << 16         'Set pins 16-19 to be outputs
outputs       long      %0000
_speed1       long      0       'Place to store central Hub RAM address of "speed1" variable on this Cog's RAM
_speed2       long      0
_speed3       long      0
_speed4       long      0
timer         res
LoopCounter   res
duty1         res
duty2         res
duty3         res
duty4         res
state1        res
state2        res
state3        res
state4        res
StartTime     res
              fit
```

Figure C3.2 – Spin code loaded to EEPROM on the final assembled Rotundaur. With the final assembled Rotundaur and perfboard, this code is currently nonfunctional despite being a combined version of the function wheel and puppet mode arm codes. It is theorized that there is a small typo between versions causing the errors or miswiring on the perfboard.