Proceedings of The National Conference
On Undergraduate Research (NCUR) 2016
University of North Carolina Asheville
Asheville, North Carolina
April 7-9, 2016

A Case Study in Autonomous Robot Design

Jeffery Hay, Ross Heninger, Stephen Moyer, Casey Murphey,
Christian Newhan, Fiona Popp, Corey Pullium, and Kyle Ward
Engineering - Mechatronics
The University of North Carolina Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisor: Dr. Rebecca Bruce

Abstract

The IEEE (Institute of Electrical and Electronic Engineers) Southeast Conference (SoutheastCon) Hardware Competition was created to bring together undergraduate students from universities across the Southeast region to test their technical knowledge and skills in a friendly robotics competition. The 2016 challenge was to develop a robot that would autonomously operate as a mobile shipping container crane. This challenge incorporated navigation on a game board (a 'shipyard') while collecting blocks ('shipping containers') and distributing them to various areas of the board (to a 'boat,' a 'truck', and a 'train') based on the block's location, size, and color. In order to generate unique solutions to the competition design challenges, research was done on robot mechanics, sensors, motors, motor drivers, microcontrollers, and communication between embedded devices. With respect to the focused research that was undertaken, sensors were particularly important. The UNC Asheville robot, named Ripley, incorporated sensors for echolocation, light intensity and reflection, infrared proximity, depth, torque, and machine vision. Meeting the 2016 SoutheastCon Hardware Competition challenge required a systematic approach to engineering design including research, development, and testing under budget and time constraints.

Keywords: Mechatronics, Robotics, SoutheastCon

1. Introduction

The hardware competition is one of many student competitions held at the annual IEEE Region 3 Conference. SoutheastCon is also a place where professionals gather to share technical research papers and informational sessions. The competition was designed to offer the opportunity for students to design and develop autonomous robots incorporating facets of engineering across the board; specifically electrical, electronic, mechanical, materials, computer, and controls engineering. The challenge criteria is defined such that the student teams are able to complete the design and fabrication of a robot within roughly an eight month period under the constraints of technical aptitude, budget, and time allotted outside of normal coursework. The competition teams at UNC Asheville are developed and managed by the UNC Asheville student branch of IEEE, under the guidance of faculty within the UNC Asheville engineering department as well as the Western North Carolina section of IEEE.

Traditionally, the hardware competition teams are formulated such that various engineering disciplines are represented within a team in order to successfully develop an interdisciplinary engineering solution to the challenge. However, UNC Asheville's team is highly unique. Students that make up the team are not diversified by their individual engineering disciplines, but rather unified by a common discipline, Mechatronics. The undertaking of this year's challenge by a team of Mechatronics engineering students demonstrated the advantages of this unique engineering approach.

2. The 2016 Hardware Competition Challenge

The 2016 challenge was to autonomously navigate a gameboard (Figure 1), collect blocks at three different heights (Figure 2), and distribute these blocks to three different locations (Figure 3) based on the blocks placement on the gameboard, as well as the size and color of the blocks. The gameboard was also split into two different orientations, one being a mirrored version of the other. The robot needed to know which orientation of the board it was currently playing on by either a switch that could be tripped prior to the start of the competition round or by sensing its environment prior to gameplay. The size criteria for the robot was to fit within a twelve inch cube at the start, with the ability to expand up to a twenty inch cube during gameplay.

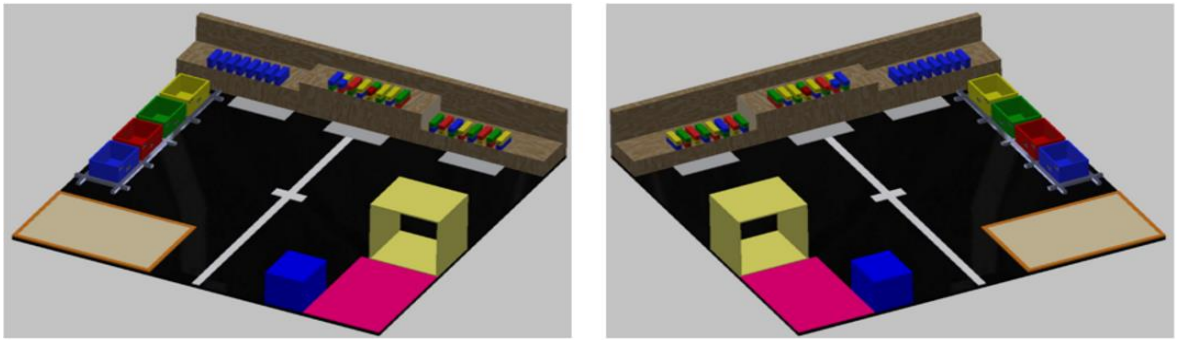

Figure 1. Both versions of the 2016 gameboard.

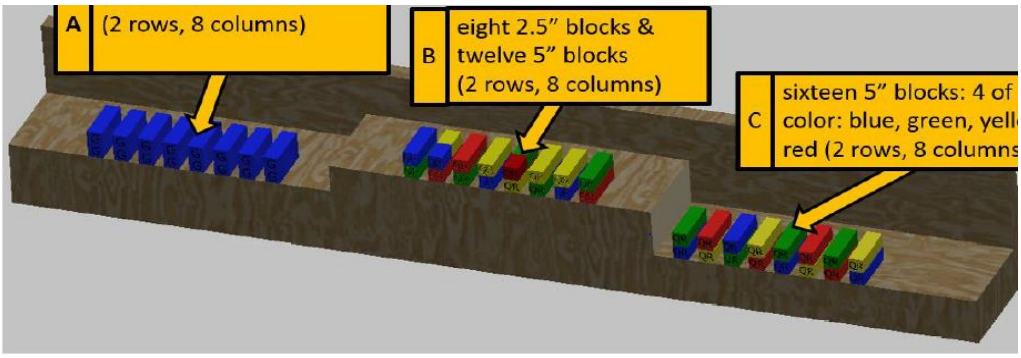

Figure 2. The shipping container barge.

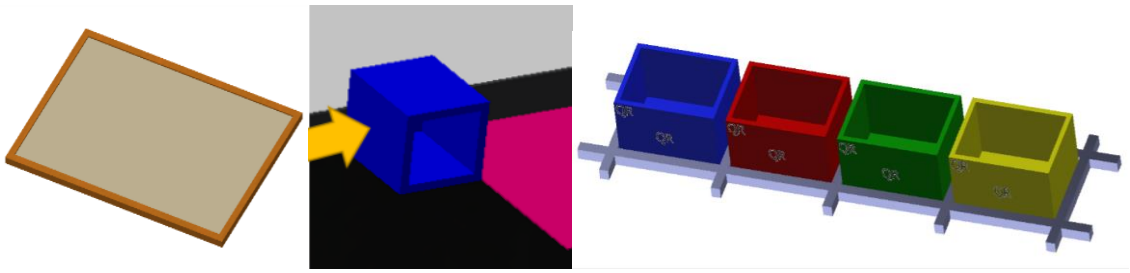

Figure 3. Left to right: The boat, the truck, and the train cars.

2.1. Navigation

In order for the robot to navigate the gameboard autonomously, it needed a way to sense its orientation relative to its environment and a way to traverse the gameboard from location to location. Six inch high walls surrounded the

gameboard which provided a means for the robot to sense its two dimensional orientation relative to the walls wherever a line-of-sight existed. Careful research was done in order to determine all possible options for navigation sensing [2], and a decision matrix was developed in order to choose the most appropriate method (Table 1). The sensing method determined was binocular echolocation by PING sensors. These sensors operate by sending a high frequency pulse from one speaker (a 'chirp') and receiving the pulse back from the other speaker (an 'echo'), monitoring the time elapsed, and calculating a distance based off of the time and initial calibration. The reliable range for the measurements is two centimeters to three meters. The reliable offset angle between the sensor and the object being detected is fifteen degrees. Given these constraints, and operating in a temperature controlled environment, PING sensor measurements are accurate to within ten percent of the actual distance.

Table 1. The decision matrix for navigation sensing

NAVIGATION SENSORS								
	0-5	*0-10*	*0-10*	*0-10*	*0-5*	*0-10*	*0-10*	*Scale*
Component	Ease of Physical Integration	Simplicity	Low Cost	Closed Loop	Self Processing	Accuracy	Multi-Purpose	**Total**
Laser RangeFinder	4	1	2	5	5	10	0	27
MAX Sonar	5	5	4	5	5	8	0	32
Parallax PING	**5**	**8**	**6**	**5**	**2**	**7**	**0**	**33**
PIXY Camera	2	2	3	5	5	6	5	28
9 DOF IMU	5	4	8	0	3	8	0	28

In order for the robot to traverse the gameboard, it needed a complete chassis drivetrain. Choosing from many existing options for implementing a drivetrain, careful consideration was given to determine the most appropriate method (Table 2). The wheels that were chosen, mecanum wheels, are unique in that they can be oriented and driven the same as traditional wheels, but also driven in a manner that allows for holonomic movements due to the angled roller bearings that surround the wheel.

Table 2. The decision matrix for navigation wheels

NAVIGATION WHEELS							
	0-5	*0-10*	*0-10*	*0-10*	*0-10*	*0-10*	*Scale*
Component	Ease of Physical Integration	Weight Bearing Capacity	Frictional Integrity	Rigidity	Holonomic Ability	Cost	**Total**
Tracks	2	8	10	7	0	5	32
Standard Rubber	5	7	6	6	0	10	34
Omniwheel (TRI)	3	5	4	7	5	8	32
Omniwheel (QUAD)	4	6	5	7	10	8	40
Mecanum	**5**	**7**	**7**	**6**	**10**	**7**	**42**

The decision matrix process was repeated in order to chose the most appropriate motor to drive the wheels (Table 3). The DC brushed motor chosen is a traditional gearmotor that operates by energizing a coil winding, which in turn forces the output shaft magnet to rotate in the direction opposite of the current flow. To drive the motors at a high voltage and current with a microcontroller, standard H-bridges were implemented which allowed low level logic signals to directly drive high level outputs. H-bridges are built out of four transistors, and the direction of current flow through the motor is the product of opening and closing the transistor gates by means of logical input signals.

The speed of the motor can be varied by pulse width modulating the logic signal on the high potential gate being opened within the H-bridge circuit. Pulse width modulating is a form of digital control that creates an average power supply to a load that can be varied between zero and full supply, with a resolution of 255 values in between the two extremes.

Table 3. The decision matrix for navigation motors

NAVIGATION MOTORS						
	0-5	*0-10*	*0-10*	*0-10*	*0-10*	*Scale*
Component	Ease of Physical Integration	Torque / Power Consumed	Driver Options	Complexity	Cost	**Total**
DC Brushed	**4**	**6**	**8**	**7**	**8**	**33**
DC Stepper	5	7	8	4	5	29
Continuous Servo	3	5	10	6	7	31

2.2. Manipulation

In order for the robot to move the blocks from the barge to their perspective destinations, a manipulator arm and gripper assembly was needed. An abundance of options existed for how this challenge might be overcome in design. The preliminary stage of design was to outline exactly what the manipulator arm and gripper needed to accomplish under the constraints of structural integrity and weight bearing capacity at all positions required by the challenge. First, the arm and gripper assembly needed to be compact enough to fit within the starting position footprint. Second, the arm needed to be able to maneuver the gripper assembly to all possible locations where the blocks would be collected and distributed. Third, the gripper assembly parameters needed to be determined such that the maximum amount of blocks could be collected without hindering the prior constraints in order to complete all of the tasks within the allotted time period of five minutes per round.

Careful research into multi-dimensional robot arms indicated that the axes should consist of a combination of rotational and prismatic (translational) joints, the latter being the less computationally expensive in terms of positioning via feedback control [6]. In industrial robotics, the design of a manipulator should encompass all complexities required with as simple of an implementation as possible. From this research, it was determined that using as few prismatic joints as possible would allow for the least complex programming and fabrication while still providing a complete solution to the manipulator arm challenge. The manipulator arm needed a Z-Axis prismatic joint such that the total assembly could fit within the starting footprint and be maneuvered to reach forward so that the gripper assembly could hover over the blocks. Dual Y-Axis prismatic joints were needed such that the total assembly could be lifted to a height above the highest blocks on the barge, and lowered in order to fit within the truck destination. Dual X-Axis prismatic joints were needed such that the total assembly could reach the far left and right extremes where blocks were located that the chassis alone could not reach (Figure 4).

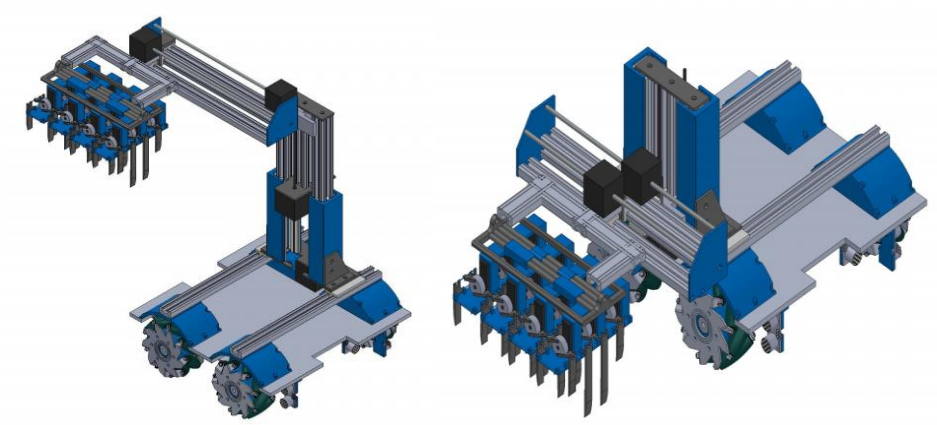

Figure 4. The manipulator arm positions in order from the chassis attachment point. Left: Z-rear, Y-top, Y2-top, X-right, X2-right. Right: Z-forward, Y-bottom, Y2-bottom, X-center, X2-center.

Once the determination was made on the core design of the manipulator arm, five prismatic degrees of freedom, the method for developing the linear motion of each axis was decided by a decision matrix (Table 4).

Table 4. The decision matrix for manipulator arm methodology

MANIPULATOR ARM METHODS							
	0-5	*0-10*	*0-10*	*0-10*	*0-10*	*0-10*	*Scale*
Component	Ease of Physical Integration	Weight Bearing Capacity	Frictional Integrity	Rigidity	Compactability	Cost	**Total**
Rack and Gear	2	4	9	8	6	8	37
Lead Screw	**5**	**7**	**7**	**6**	**8**	**7**	**40**
Linear Slide	3	4	6	5	10	8	36
Belt and Pulley	2	5	5	4	6	6	26

Another critical aspect of the manipulator arm design was sensing the position of the arm and gripper assembly relative to the robot chassis. The endpoints of each axis needed to be sensed in order for the safety of the joints and maintaining the integrity of their relative positions. A decision matrix was developed in order to the determine the most appropriate method for sensing all linear positions of the manipulator arm (Table 5).

Table 5. The decision matrix for manipulator arm position sensing

MANIPULATOR ARM SENSORS							
	0-5	*0-10*	*0-10*	*0-10*	*0-5*	*0-10*	*Scale*
Component	Ease of Physical Integration	Complexity	Cost	Closed Loop	Processing Demand	Accuracy	**Total**
IR Proximity	4	8	5	0	4	7	28
Limit Switch	**5**	**9**	**8**	**0**	**5**	**9**	**36**
Reflectance	2	4	5	5	2	6	24
Linear Potentiometer	3	6	7	5	3	8	32

In order for the robot to move the blocks once the manipulator arm was positioned, grippers were needed. In industrial robotics, various gripping designs are implemented depending on the application. For our challenge, we chose pinching style grippers. Since the rotational range required to open and close a gripper on a block would not exceed thirty degrees, non-continuous servo motors were an easy choice without the need for a decision matrix.

To position the gripper assembly on the blocks, three axes of precision were needed. The Z-axis positioning was accomplished by the navigation sensors lining up on the barge. The X-axis positioning was accomplished using a bank of infrared proximity sensors embedded within the gripper assembly that located the gaps between the block stacks. The final Y-axis positioning was determined by button contact switches closing when the gripper sets made contact with the tops of the block stacks.

When the blocks were picked up by the grippers, the robot needed a way to know the color of each block it held. A variety of color sensing options were explored (Table 6) and it was determined that the most advantageous method for color sensing was by integrating a sensor into every gripper module.

Table 6. The decision matrix for block color sensing

COLOR SENSORS							
	0-5	*0-10*	*0-10*	*0-5*	*0-10*	*0-10*	*Scale*
Component	Ease of Physical Integration	Simplicity	Low Cost	Self Processing	Accuracy	Multi-Purpose	**Total**
Parallax ColorPal	3	4	5	2	8	5	27
TCS34725 RGB	4	6	8	5	5	0	28
Flora RGB	**5**	**6**	**8**	**5**	**5**	**0**	**29**
PIXY Camera	2	3	2	4	7	10	28

3. A Mechatronics Design Approach

With the challenge broken down and considering all possible conceptual design approaches as well as choosing methods for developing solutions, a complete design implementation was realized. The Mechatronics design approach can be broken into the discrete categories of mechanical, electrical, and programming. The UNC Asheville team of

Mechatronics engineering students was divided into these separate disciplines, though keeping the Mechatronics mindset at the front of the design process in all aspects of development which allowed for more fluid adjustments within and between the subset teams. Continuous collaboration as a whole promoted the Mechatronics experience, and focus within the subset teams provided a means to divide the workload such that catering to the strengths of the individual could be promoted as well. In order to aide the development phase, a variety of design tools were utilized (Table 7). All of these tools were either provided by the institution or available as free and open-source software.

Table 7. Design tools used for the development of the robot

DESIGN TOOLS UTILIZED DURING DEVELOPMENT	
Software	**Purpose**
Solidworks CAD	To model the robot chassis and arm mechanics, develop blueprints for CNC Machining, and develop sub-assemblies for 3D printing
Fusion 360 CAD	To model the robot gripper assembly mechanics and develop sub-assemblies for 3D printing
Eagle CAD	To build and organize circuit schematics, and design printed circuit boards for fabrication
Arduino IDE	To write and compile the hardware processing code to be flashed onto the microcontrollers
Processing IDE	To develop GUI interfaces for modifying microcontroller parameters dynamically, specifically for the master processor
TextWrangler	To write and edit object oriented C++ files and header files to be included on the arduino code compilation for the microcontrollers
Digikey SketchUp	To build and refine the component interconnectivity roadmap, and provide a means for organzing the total electrical architecture
Lucid Chart	To flowchart the logical decision-making by the microcontrollers to fully describe the total processing dataflow
GitHub	To develop and manage code revisions progressively through the use of private repositories

3.1. Mechanical Design

The mechanical functionality of Ripley was determined by careful decision matrix formulation to consist of a mecanum-wheeled chassis, a five degree of freedom manipulator arm, and a gripper assembly composed of twelve independent gripper mechanisms.

The requirements of the robot chassis were outlined such that it could support the total weight of the manipulator carrying a payload with a minimum bending stress imposed during gameplay. The chassis also needed to house and protect the major control system componentry, power supply, navigation sensors, and drivetrain. Options considered for the material composition of the chassis were hardwood, steel, and aluminum. Minimizing weight while also maintaining structural integrity lead the mechanical team to choose cast aluminum for the chassis frame. The wheel hub supports, motor mounting brackets, and manipulator arm attachments were also determined to be made from cast aluminum due to the material's strength and lightweight metal properties. In order to develop a specification for the mecanum wheels, a weight maximum was determined for the manipulator assembly to be sixty kilograms, which was predicted to be over twice the actual weight of the robot. This lead to the choice of 100 millimeter aluminum alloy wheels with rubber rollers. Custom metal hubs were needed to connect the wheels to standard gearmotor shafts, and were CAD designed to be CNC machined, along with the wheel hub supports [4].

In order to maximize the space allotted by the robot footprint, the underside of the chassis needed to house as many of the components as possible, including as much of the weight as possible in order to keep the robot's center of mass low to the ground. It was determined that the underside of the chassis would house the motors, motor brackets and hubs, wheels, navigation sensors and processor boards, the power supply, and all voltage regulators. The top of the chassis also needed to be carefully organized in order to house the manipulator mounts on top of the wheel hubs, main processor board, motor driver boards, any additional processor boards not applicable to the gripper assembly, and the main wiring harness connection to the gripper assembly (Figure 5).

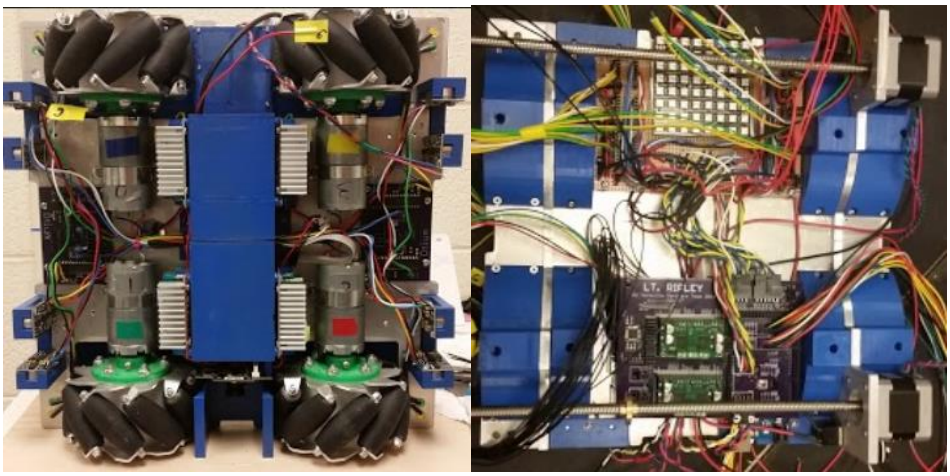

Figure 5. Left: The underside of the robot chassis. Right: The top of the robot chassis.

Similar to many modern additive 3D printers, the manipulator arm consisted of motorized lead screws in order to maneuver in five degrees of freedom prismatically. Starting from the attachment to the chassis, the Z-axis supported all components above its attachment, was dual-rail mounted to the chassis wheel hubs, and was driven by two logically tied stepper motor lead screws which provided the forward and backward motion of the manipulator assembly relative to the chassis. From this joint, the main Y-axis supported all components above its attachment, was centrally mounted on the Z-axis attachment, and was driven by a single stepper motor lead screw which provided the main up and down motion of the manipulator assembly relative to the chassis. An additional Y-axis, similar to the former, was mounted to the forward face of the former which provided a secondary up and down motion for the manipulator assembly relative to the chassis. From this joint, the dual X-axis assembly was mounted, also operated by stepper motor lead screws. The secondary X-axis was in front of and above the primary X-axis, which provided a left and right motion, with extension, of the gripper assembly relative to the chassis. All joints of the manipulator arm, attached to lead-screw mounts, were supported by linear friction rails when travelling from one end extreme to the other and made from t-slotted aluminum framing.

The most carefully designed component of the robot was the gripper assembly. This assembly, as specified in the conceptual design, was required to house four sets of three individual gripper mechanisms, a linear collapsing mechanism for each set, and all motors, sensors, microcontrollers, and cabling required by the assembly with only a communication and power supply harness running down the arm to the chassis. Careful considerations were given to the routing of cables from all sensor and motor components to the processor board such that the components could be serviced or replaced if necessary (Figure 6).

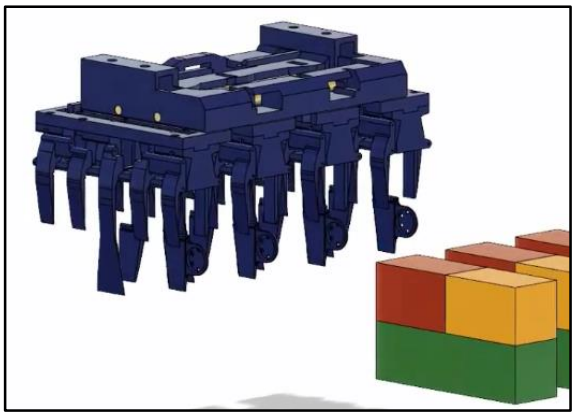

Figure 6. The complete gripper assembly, shown next to and above stacks of blocks.

3.2. Electrical Design

With twenty-five motors, ten motor drivers, over fifty sensors, and ten microcontrollers, proper electrical design was crucial to the overall development of the robot's functionality. At the center of the control system was an Arduino Mega which managed all major dataflow of the robot. Coupled to this processor was a PCB (printed circuit board) which contained all major wiring harness connections to external processor boards and stepper motor drivers, as well as the drivetrain motor drivers (Figure 7) [5].

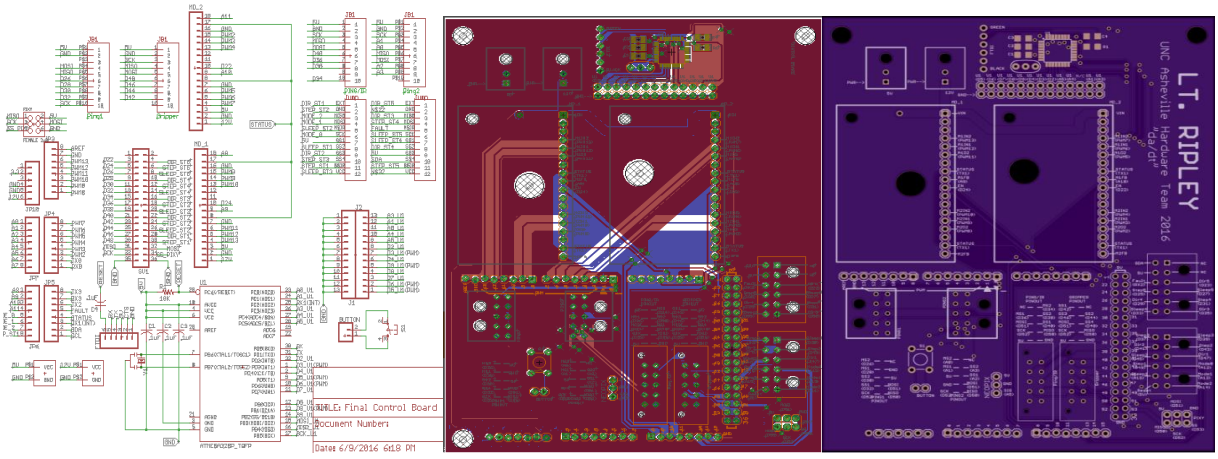

Figure 7. The master control board. Left to right: The schematic, board design, and finished PCB.

On the underside of the chassis was two dual-processing navigation PCBs, which took sensor data from the PING sensors to feed the internal PID (proportional, integral, derivative) control navigation of the robot computed by the master processor. On top of the chassis was a single thread processor that took sensor data from the limit switch contacts to feed the manipulator arm movements computed by the master processor. On the gripper assembly was four independent processors combined onto one PCB, one for each set of grippers, that took commands from the master processor in order to feed their own internal PID control positioning and gripping actions. The input to the gripping controllers were current sensors from the servo motors actuating the individual gripper mechanisms, internally calculated as a torque imposed on each container by the gripper fingers.

All microcontrollers on the robot operated at five volts, as well as the majority of the sensors (three volt logic and power supply was used for the color sensors). The drivetrain motors required twelve volts, and the stepper motors fifteen volts. The servo motors on the gripper assembly operated at six volts. All motors used five volt logic from their drivers communicated by their perspective microcontrollers. A power analysis yielded the need for four regulated voltages, each with their own current demands, which aided the specification for the power supply (Table 8).

Table 8. Power analysis and supply specification

POWER SUPPLY SPECIFICATION							
	Servo Motors	DC Drive Motors	DC Stepper Motors	Processors	Sensors/Drivers	**Total Watt-Hours**	**Total Amp-Continuous**
Operating Voltage	6	12	15	5	5	39.3	24
Current Draw (Amps)	0.25	3	4	0.25	0.05		
Time Required (Hours)	0.06	0.09	0.06	0.12	0.12	(For 12V Battery)	
Amount (Multiplier)	15	4	6	10	60	**Total Amp-Hours**	**(For 12V Battery) Minimum Specifications**
Watt-Hours Required	**1.4**	**13**	**21.6**	**1.5**	**1.8**	3.3	**12V, 5Ah, 25A Cont**

The single supply chosen was a NiMH (nickel metal hydride) battery with twelve volt potential, a ten amp-hour life cycle, and a maximum thirty amp continuous current draw. The battery voltage was bucked/regulated to the needed twelve volts, six volts, and five volts, and boosted/regulated to the needed fifteen volts.

Inductive and capacitive analysis was performed on all processor boards, motors, and drivers in order to determine the proper discrete filtering circuitry required to ensure that logical connections were undisturbed, and power fluctuations managed throughout the robot's operation [7]. Decoupling capacitors on power rails and driver circuits, in-line capacitors on high pressure, low current logic circuits, and voltage dividers were implemented in order to meet the requirements produced in the analysis. The choice of universal connectors was Molex MicroFit, which allowed for secure cable connections between the embedded devices, custom harness braiding, serviceability of individual components, and a less expensive alternative to pre-made harnesses.

3.3. Programming Design

With all of the componentry outlined in the electrical section, it is easy to assume a high level of complexity within the programming side of Ripley. All microcontrollers used were single-thread, with varying amounts of memory depending on the demand from each device in what it needed to calculate, communicate, and logically manipulate. Beginning with the master processor, all major decisions and dataflow operations were organized such that a series of steps were taken throughout gameplay in order to accomplish the many processing tasks. Thousands of lines of code, written in the Arduino native language (C++), required proper management [3]. Incremental testing and proper attention to safety allowed the programming team to successfully develop the logical decision-making ability of Ripley. The autonomous orchestration of ten microcontrollers demanded precise timing and a robust communication protocol. Ripley's predominate form of communication was Serial Peripheral Interface (SPI). SPI is a common communication method used in embedded systems due to its speed, lightweight hardware, and flexibility.

The navigation portion of the code consisted of the navigation slave processors taking in data from the echolocation sensors and passing the information to the master, the master calculating the proper PID correction based on the chassis' orientation relative to the gameboard, and progressively driving the chassis to the current destination [1]. This was repeated for every movement of the chassis throughout gameplay from coordinate to coordinate on the board, depending on the current step in the overarching dataflow process. The arm positioning portion of the code consisted of the arm slave processor taking in data from its limit switches, and passing the information to the master. The master would then calculate the proper PID correction based on the grippers orientation relative to the gameboard, and would progressively drive the manipulator arm to the correct destination.

The boat-distributing portion of the code consisted of the master navigating the chassis to its proper destination and commanding the gripper slave processors to drop all containers. The truck-distributing portion of the code consisted of the master commanding the gripper slave processors to collapse the entire assembly, feed information from the arm slave processor to the master for position relative to the truck, and navigating into the truck before commanding the gripper slave processors to drop the necessary containers.

The traincar-distributing portion of the code incorporated machine vision into Ripley's strategy. The master utilized information fed from the PIXY camera located on the underside of the chassis. The containers were distributed by navigating along the train cars, computing PID corrections based on the navigation sensors and cameras pixelated color data, and commanding the gripper slaves to drop the containers when necessary. Due to the high level of complexity, all functionalities of the robot were tested independently before collating into the final code segments.

4. Conclusion

The 2016 IEEE Southeast Conference Hardware Challenge offered the opportunity for undergraduate engineering students to work through the engineering design process as a team, develop complex solutions with a high degree of technical aptitude, expand an understanding of what it takes to design and fabricate a finely specified robot with heavy budget and time constraints, and further develop skillsets with real application. All aspects of the 2016 challenge were highly relevant to what Mechatronics is all about, tackling real world problems and developing solutions to those problems through an interdisciplinary methodology.

Dividing down the responsibilities of the team into core mechanical, electrical, and programming groups aided the overall success of fabricating and testing the functionalities of the robot. Although the time constraint did not allow for completion and the UNC Asheville team's attendance at the conference, there was no loss in the value of the project as a whole with such a tremendous amount of experience gained in all aspects. The finished robot was presented and demonstrated at the NCUR 2016 Symposium at UNC Asheville (Figure 8).

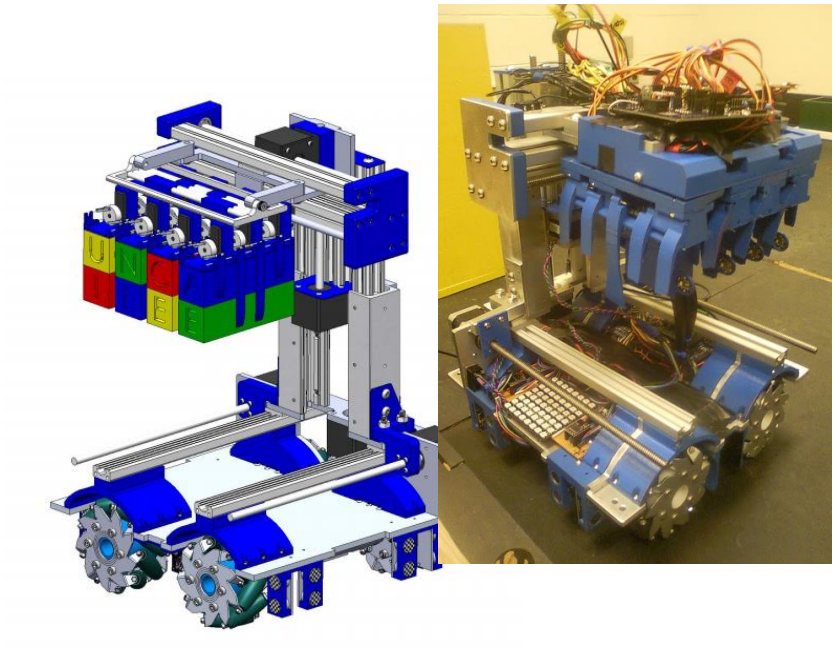

Figure 8. Left to right: The CAD model of the robot and the finished robot.

5. Acknowledgements

The authors wish to express their appreciation to the UNC Asheville engineering department for all of their support throughout the team's development phases. The authors also wish to express their appreciation to OSH Park for their continued support and providing the team with PCB manufacturing services.

6. References

[1] Burzo, Emil. *Pid Control: New Identification and Design Methods*. S.l.: Springer, 2010.

[2] Fraden, Jacob. *Handbook of Modern Sensors: Physics, Designs, and Applications*. New York: Springer, 2010.

[3] Lippman, Stanley B., Josée Lajoie, and Barbara E. Moo. *C Primer*. Upper Saddle River: Addison-Wesley, 2013.

[4] Lombard, Matt. *SolidWorks 2013 Bible*. Indianapolis, IN: Wiley, 2013.

[5] Monk, Simon. *Make Your Own PCBs with EAGLE: From Schematic Designs to Finished Boards*. New York: McGraw-Hill Education, 2014.

[6] Ross, Larry T., and James W. Masterson. *Robotics: Theory and Industrial Applications*. Tinley Park, IL: Goodheart-Willcox Company, 2010.

[7] Scherz, Paul, and Simon Monk. *Practical Electronics for Inventors*. New York: McGraw-Hill Education, 2015.