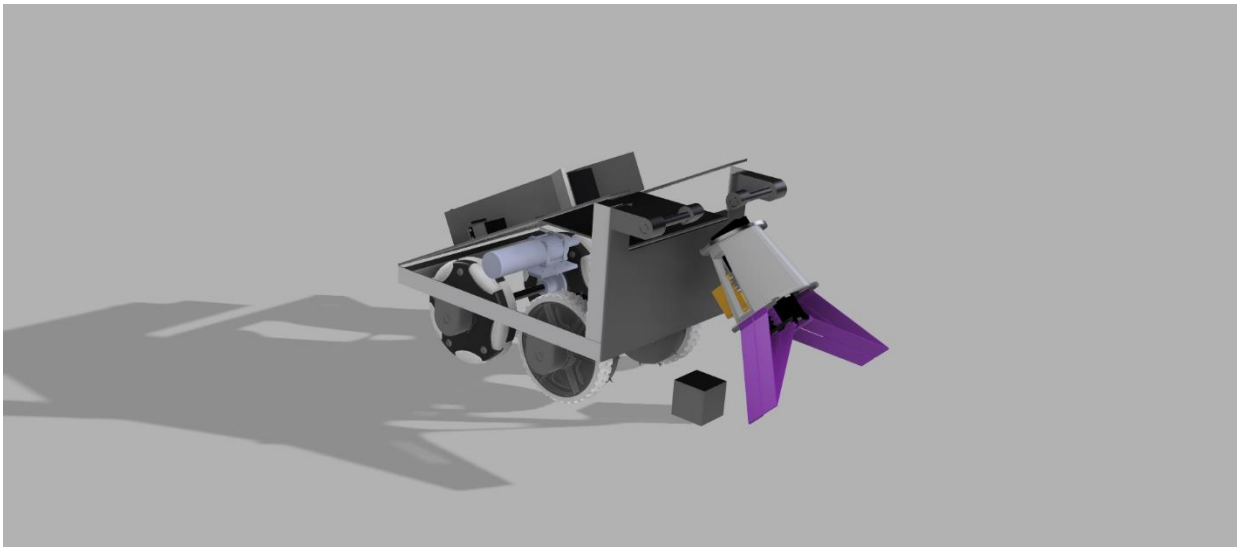
2016 ATMAE Robotics Technical Report



East Carolina University

This document contains the technical information, explanation, and justifications for East Carolina University's 2016 robotics build.

Acknowledgements

We would like to thank Dr. Tijjani Mohammed, Ms. Amy Frank, and Dr. Jimmy Linn for their dedicated support throughout the duration of this project. We would also like to extend gratitude to William "Bill" Mcclung for his knowledge and contributions towards the project.

East Carolina Chapter of ATMAE

East Carolina University
Greenville, North Carolina

College of Technology and Computer Science Department Head

- Dr. Tijjani Mohammed MOHAMMEDT@ecu.edu

ATMAE Chapter Advisor

- Ms. Amy Frank FRANKA@ecu.edu

Robot Project Faculty

- Dr. Jimmy Linn LINNJ@ecu.edu

ATMAE Cabinet

- Eric Panarusky: President
- Tyrus Keen: Vice-President
- Chance Smitherman: First Officer

ATMAE Student Build Team

- Zack Cleghorn: Team Captain
- Joshua Stevens: Designer
- Jordan Hughes: Technical Report Editor
- Chance Smitherman: Electronics
- Cameron Coleman: Fabricator
- Luke Pearson: Lead Programmer
- Sam Saunders: Fabricator
- David Palmieri: Programmer
- Huan-Yun Chen: Programmer
- Nicholas DeBella: Programmer

Table of Contents

Executive Summary

This year is another exemplary year for East Carolina University students to shine at the ATMAE robotics competition. With this year's them being the same as last year; our team has decided to reengineer our robot. Every aspect of last year's robot from East Carolina University was scrutinized. Our team decided to move with a new robot, instead of reusing last year's design. After an extensive analysis, it was decided that a new platform was necessary. Further explanation of the design will be explained later in the report.

This year also marks a change in our robotics team. We have before always had at least one person from last year's competition on the team. However, with all the previous students graduating; This robotics team is completely new and fresh. This new robot team comes with dedicated students. Time was the most critical factor of our project. We assembled a team September 1st, and then started working on the project.

Despite all of the challenges, our team has built a product that we are extremely proud of. It is our hope that the current robot will be a foundation for future teams at East Carolina to use in whatever competitions they choose to be a part of. Although this project is still fairly new to our university, the students at East Carolina are very excited to be a part of this experience, and to represent our school at the national conference.

Project Timeline

Task Name	Duration	Start	Finish	% Work Complete	Remaining Duration
Robot Fully Completed	**48 days?**	**Mon 8/22/16**	**Wed 10/26/16**	**0%**	**10.59 days?**
Design	17 days?	Mon 8/22/16	Tue 9/13/16	100%	0 days?
Chassis	18 days?	Tue 9/13/16	Thu 10/6/16	100%	0 days?
Fabrication	22 days?	Tue 9/13/16	Wed 10/12/16	0%	4 days?
Testing	11 days?	Wed 10/12/16	Wed 10/26/16	15%	11 days?
Wiring/Hardware	6 days?	Wed 10/12/16	Wed 10/19/16	5%	6 days?
Programming	1 day?	Wed 10/19/16	Wed 10/19/16	25%	1 day?
Trial/Testing	14 days?	Wed 10/12/16	Mon 10/31/16	25%	14 days?
Autonomous Path Following	6 days?	Wed 10/12/16	Wed 10/19/16	10%	6 days?
Burlap Climb Obstacle	8 days?	Mon 8/22/16	Wed 8/31/16	100%	0 days?
Poster	1 day?	Wed 10/26/16	Wed 10/26/16	15%	1 day?
Technical Report	34 days?	Mon 8/22/16	Thu 10/6/16	100%	0 days?

*Note- This is the most recent updated project file prior to the completion of the technical report. 100% completion is assumed at the time of the competition.

Project Costs

Part	Serial Number	Price	Qty	Total
Adafruit 16-channel 12-bit PWM interface	815	$ 14.95	3	$ 44.85
Bus Board	538-38770-0108	$ 3.87	4	$ 15.48
Gear box	P60K-444-0004	$ 64.50	2	$ 129.00
ADC 8 Channel Analog to Digital MC (MCP3008)	856	$ 3.75	2	$ 7.50
Motor Controller	B00YZV80C0	$ 5.99	2	$ 11.98
Motor	KS-550-7527	$ 7.50	2	$ 15.00
		Total		$ 223.81

Printed Part	Weight in Grams	Price	Qty	Total
Wheel Axle Spacer	8	$ 0.80	5	$ 4.00
Wheel Adapter	74	$ 7.40	3	$ 22.20
Wheel Bushing	1	$ 0.10	2	$ 0.20
Gripper Armature	147	$ 14.70	1	$ 14.70
Holding Support	5	$ 0.50	4	$ 2.00
				$ 43.10

TOTAL COST OF ROBOT	$ 266.91

*Note- These costs do not reflect the cost of parts donated by team members.

Robot Construction

- Design

The design for the 2016 East Carolina build is centered on speed, mobility, ease of operations, and efficiency. The primary goal of the design is speed and accuracy. Simplicity is a key aspect in deciding how to pick up and store each one of the blocks of the robot. To help with mobility, the frame must be light. We design the frame using aluminum angle-iron metal (1inch). We noticed last year's robots were very top heavy. So with this year's design, we decided to go with a wedge-type platform. Our robot sits low to the ground, to prevent tipping. The robotic arm was design for simplicity. We decided a fish-tail gripper would be the most creative aspect of our robot. We 3-D printed every part of the gripper. This was to give the judges something to be impressed by. Instead of buying parts or a kit, we created a design from scratch. The drive train consist of drill motors. This was done for two reasons: (1) to provide speed (2) powerful torque. The storage for the ATMAE blocks was created based off of a pickup truck.

- Power Source

At the moment there are two power sources. The power sources for the motors is an 18-volt nickel-cadmium (NiCd) battery that was originally meant for power tools. For the "brain" (raspberry pi) we are using a pocket juice. The pocket juice consists of a lithium ion battery with a capacity of 2000 mAh. This is fed through micro USB for connection to the raspberry pi. We are using two batteries because the motors require a large voltage source. We are also using them so that if the battery for the motor control dies the robot wont.

Capacitor Specifications

Electrical

- Rated Capacitance	500 F
- Minimum Capacitance Initial	500 F
- Maximum ESR_{DC} Initial	2.1 mW
- Rated Voltage	16 V
- Absolute Maximum Voltage	17 V
- Maximum Continuous Current ($\Delta T=15^oC$)[2]	100 A_{rms}
- Maximum Continuous Current ($\Delta T=40^oC$)[2]	160 A_{rms}
- Maximum Peak Current, 1 second (non repetitive)	2,000 A
- Leakage Current Maximum	5.2 mA
- Maximum Series Voltage	750 V

Temperature

- Operating Temperature (Ambient temperature)	
- Minimum	- 40°C
- Maximum	65°C
- Storage Temperature (Stored uncharged)	
- Minimum	- 40°C
- Maximum	70°C

Physical

- Mass, typical	5.51 kg
- Power Terminals	M8/M10
- Recommended Torque – Terminal	20/30 Nm
- Vibration Specification	SAE J2380
- Shock Specification	SAE 2464
- Environmental Specification	IP65
- Cooling	Natural Convection

Power & Energy

- Usable Specific Power	2,700 W/kg
- Impedance Match Specific Power	5,500 W/kg
- Specific Energy, E_{max}	3.2 Wh/kg
- Stored Energy	17.8 Wh

Life

- High Temperature	1,500 hours
- Capacitance Change	20%
- ESR Change	100%
- Room Temperature	10 years
- Capacitance Change	20%
- ESR Change	100%
- Cycle Life	1,000,000 cycles

-Drive Train

We are using a 2 KS-550s to power each wheel independently. This in turn will allow the robot to steer in a tank-like matter. We are using one spur gear per wheel to supply a 1:1 turn ratio. The motor is attached to a 64:1 gear box to provide a large amount of torque to the drive train. The rear axle is connected solidly to both wheels and will allow turning using Omni-wheels. From there it will allow the robot to rotate at 360 degrees by rotating each wheel in opposite directions. At maximum efficiency the motor is capable of putting out 1600 rpm.

- Retrieval Apparatus

After reviewing the 2015 competition results, our team noticed that there was a problem with the grippers. Simplicity was the main focus in designing the system. We concluded that a fish-tail gripper would be the best retrieval apparatus for this competition. A servo will open and close the robots arm and a servo will spin the gripper into the tail gate. Fewer components and moving parts in the retrieval apparatus equates to lower weight; thus, less force exerted on the top of the robot frame. Lightweight capabilities and strength are trade-offs in the design of the robot, and must therefore be analyzed very carefully. Almost all of the retrieval apparatus was printed on the 3D printer.

Programming

- Xbox 360 Controller

An Xbox 360 wireless controller is used for input to control the robot. The robot uses skid-steering, which is normally controlled with 2 joysticks. Usually the joysticks are used to individually control the direction and speed of each motor independently. However, the Xbox controller joysticks are difficult to manually keep synchronized, that is, it is difficult to match speeds with the two joysticks. A way around this is to use the controller in another, familiar way, and change the output to match a skid-steer system. The most popular games for the Xbox game system have consistently been First Person Shooter games. In a FPS, the left joystick is used to control the forward / reverse direction and speed of travel of your character. The right joystick is used to rotate the character left or right, or “look around”. The PC Program, which was specially written and tailored for this robot was made to take input from the controller in a FPS fashion and modify the output to be used for a skid-steer system. The left and right triggers are used as manual overrides to the skid-steer system. These triggers only affect forward movement, but are variable for speed. If you want the left motor to remain stationary while the right motor spins forward, press the right trigger until the desired angle is reached. This is useful for pivoting the robot on the left or right wheel, which is impossible using the joysticks alone. Using the joysticks makes the robot pivot around the center point between the two wheels.

- Robot Program

The computer that we chose to use for our project is the Raspberry Pi Model B+ (Pi). Because of the large community of hackers and computer hobbyist the Pi has a large amount of

supported software, libraries, and program examples. Because the Pi is built on the Broadcom chipset, the Pi has many general purpose input output pins (GPIOs). The GPIOs allow us to interface with other controllers on the robot namely through pulse width modulation (PWM) and I2C. The GPIOs also allow us to sense switch states and drive LEDs. Another reason that we decided that the Pi was a good fit for our purposes is because it demands very little power (5V @ < 1A) and the PCB has a very small footprint (About the size of a cell phone).

Java was the language that we decided to work with because most of our programmers are well versed in the Java language. Another huge factor in that decision is that we found a library called Pi4J that allowed us to interface with the GPIO pins. The operating system that we are using is called Raspbian, a flavor of Linux similar to Debian. We are using Raspbian because it is the most widely supported OS for the Pi.

The way that the team decided to control the robot was with a wireless Xbox 360 controller. To do this was quite ordeal, to be honest. We first had to locate a Linux driver for the wireless receiver. We then searched for a library that would allow us to get inputs from the controller. After much digging we found a library called JSTest. It was a great fit for our purposes, because the library included an example code that printed the state of all the buttons and axis on the controller to the screen. So rather than modifying that code, we just made out our program that would read the output from that program, cut that string into smaller pieces, and then finally convert those pieces into integer values.

To drive the motors, we decided that we would push up on the left analog stick to drive the left motor forward, pull back on the left stick to drive the left motor backwards. To drive the right motor, we followed the same scheme, only using the right analog stick. In our program we convert the values that the controller is outputting into a range that is acceptable to our motor controller. We then take that number and set the PWM value of the respective pin to that number. The motor controller senses the PWM that it is receiving and drives the motor. If it is receiving zero pulses per second it does not drive that motor channel. If it senses 255 pulses per second it drives that motor channel at full speed. Anywhere in the middle and it will drive the motor channel respectively.

Next we had to drive servo motors. We followed a very similar system to the aforementioned scheme, except we are using the switches on the controller to tell the servos what we want them to do. Switches obviously differ from analog buttons because they can only be true or false. So we press one switch to move the servo in one direction, and another switch to move that servo in the opposite direction. Speed control was not a concern for the servos, so that is why this solution worked so well for the servos.

The Pi only sports four GPIOs that support PWM. To get over this limitation we purchased a board from Adafruit that interfaces through the I2C bus. The board has 16 PWM channels on it, and the pins are in the standard servo pin out configuration. Another reason why this board was a good solution for our project is because it is well supported, and has an established library and countless tutorials. The basic idea is that you identify which channel you want to set, and then set the PWM value for that channel. Using this board, we are able to also able drive a solenoid, which is in the cargo bed. When the time comes to unload the payload we drive the solenoid back, allowing the blacks to fall out. Easy squeezy.

The solution that we came up with to tackle the autonomous line following portion of the competition is to sense the tape on the floor with photo-resistor cells. So we have an array of photocells detecting light bouncing off the floor on the bottom of the robot. On the bottom of the robot there are also bright LEDs to ensure that all of the cells are receiving the same amount of light, eliminating factors associated with relying on ambient light. Through research and experimentation, we discovered the resistance values that we should expect when the robot is directly over the tape versus when the photo-resistor is over the carpet. Using an algorithm that we developed, the robot can correct itself when it goes off track, as well as turn whenever the tape changes direction. The Pi lacks an on board analog to digital converter (ADC), so reading the resistance values of these photo-resistors would be impossible without an external ADC. The ADC that we are using is the MCP3008, an ADC that supports nine channels of input, and interfaces with the Pi using GPIO pins. Again, the chip is widely used in computer hobbyist circles, so there was plenty of example code and an established library specifically for the Pi.

In order to sense a block on the tape we are using a sonar range detector made by Parallax. Again, software support for this board is widely available, so implementation of this device was quick and painless. We just had to wire up the sensor, and the example code that came with the sonar library prints out range information to the screen. Using the very same method that we used for the Xbox controller, we can take that output information, cut it up into strings, convert those strings into integer values, and BAM, we have a working sonar detector. When in autonomous mode, the robot will be able to detect a block on the tape, move the servos to pick up the block, and load the block into the payload bed on the back of the robot. To put the robot into autonomous mode, we manually drive the robot to the tape, then we will press a switch on the controller to put the robot into autonomous mode.

The Pi on the robot is a headless unit. So to ensure that we do not lose disk data we have a toggle switch that is connected to a GPIO pin. When the program detects a state change the program runs a system power off routine, allowing us to do a proper power off, rather than a hard power cycle. We also have an LED on the robot that blinks, indicating that the watchdog program is responding and that no problems have incurred. If there is an issue during run-time the LED will shine steady, indicating that the watchdog has stopped a class, and thus, all of the other classes.

Upon boot up the Pi will automatically run a bash script that we made, that will compile the programs and run the software without having to interface with the Pi with a keyboard.

- Communication Protocol

The communication for the robot is based on a Java library for the raspberry pi. In this library an Xbox controller is connected through USB and provides analog and digital input. From there, the pi reads several axes for each of the joysticks and typically one for each button. The left joystick, for example, controls two of the analog axes since it needs to read forward to reverse and left to right. The max integer for forward ideally gives a value of -32767 while the max integer for reverse would ideally give a value of 32767. However, since this joystick gives information for two planes of direction these numbers mix between a combination of the two to provide an inverse coordinate system. Left joystick, right joystick, left trigger and right trigger make up the inputs for analog. Left and right trigger each only take up one axis and run from 32767 as the standard to -32767 at full squeeze. The digital axes are read with binary with 0 being the typical state of the readout. This would include the remainder of the inputs such as a, b, x, and y buttons as well as start, XBOX, right bumper and left bumper.

In order for the pi to communicate with the library a Java program was written that takes a large readout of the current values and converts this into a string. From there a slice is taken of what values are needed and the program decides how to spit this out into the motor drive or sequence initiation. At the moment, left joystick will give an analog output with -32767 as full speed forward and 32767 for full speed reverse for the left motor. This is then repeated the same for the right motor and will allow independent control of each motor as well as provide turning for the vehicle. In order to control the armature, the left and right buttons on the Directional Pad (D-pad) has been mapped in order to raise and lower the armature. The armature will also have the ability to be controlled for the relay race for the globe pieces. Up and down on the D-pad will control the opening and closing of the grabber. This will allow easier programming and better control of the full armature for the relay race. For automation, the start button will tell the robot to being a subsequence in the program.

In the automation section, we've decided to use an ultrasonic sensor in order to detect distance from the block. Once the block enters into the necessary reach for the robot the armature section of the program will run and collect the block and then store it upon its back and continue until all five pieces are collected. In order to detect where the line is under the robot we've decided that IR sensors will allow the robot to easily navigate the line. With this a signal is given out and the robot measures the bounce back of the IR light it has emitted. The reflective nature of the aluminum tape provides a solid return of a higher signal in which the robot determines the position of the line. In a simple example of the robot, a 45° turn to the right is taken by the line. The robot will move forward until the center sensor no longer measures above

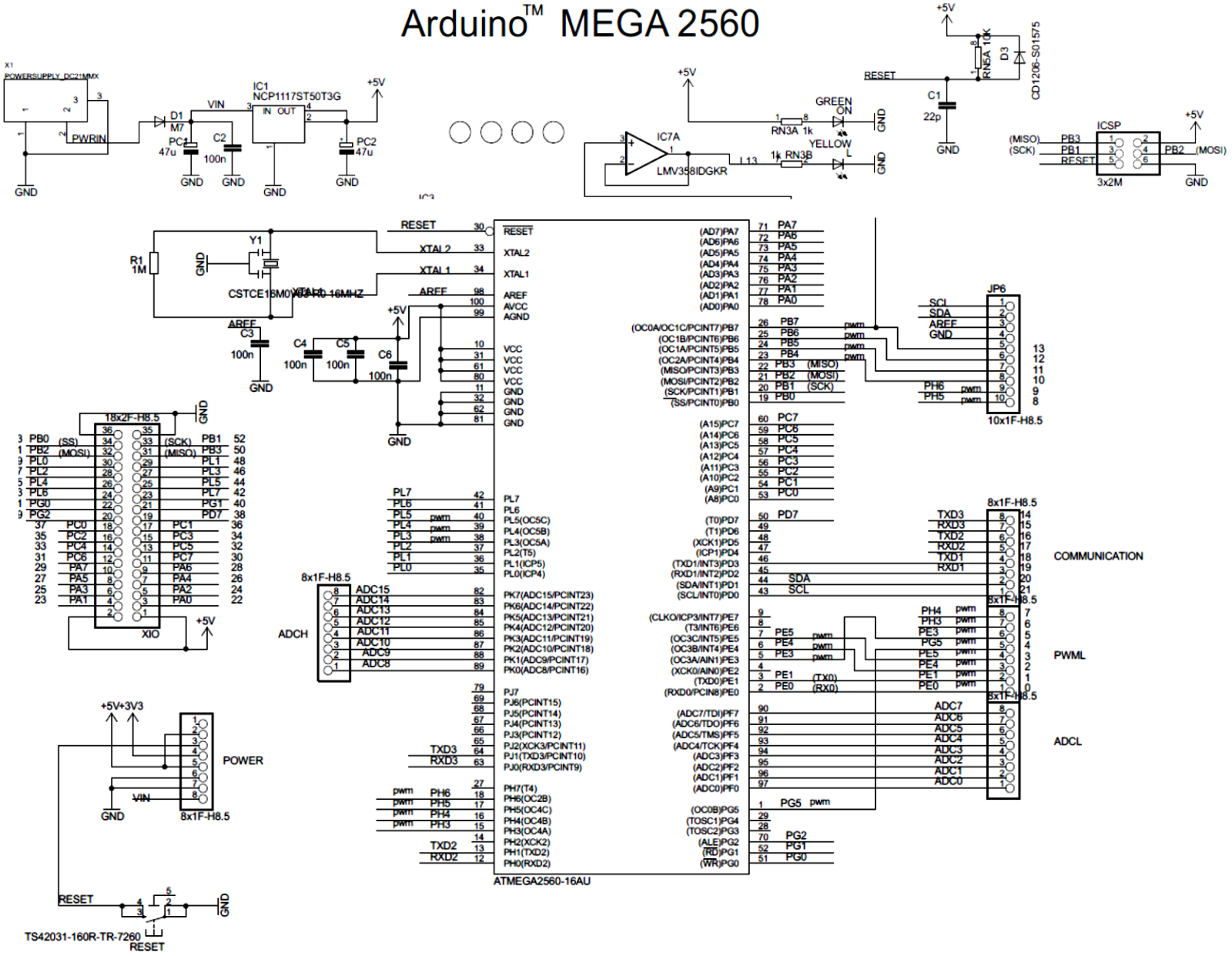
a preset threshold from the sensor. The sensor on the right will then light up and the robot will decrease speed to the right motor allowing for a turn to the right where it will continue to navigate.

Conclusion

In conclusion to this report, the East Carolina chapter of ATMAE would like to recapitulate our gratitude, and excitement in taking part in the 2016 national conference. Despite this project posing many challenges to our members, we believe it has been a valuable learning experience for everyone. Members of vast disciplines have come together to achieve something that one single member could not attain alone. Our members have come to appreciate the value of the project, as well as team work. Predominantly, our goal to establish a foundation for future teams at East Carolina was lucrative. We can now say with upmost pride, that we have achieved our objective. We look forward to seeing the other competing institutions, and exhibiting our product at the competition.

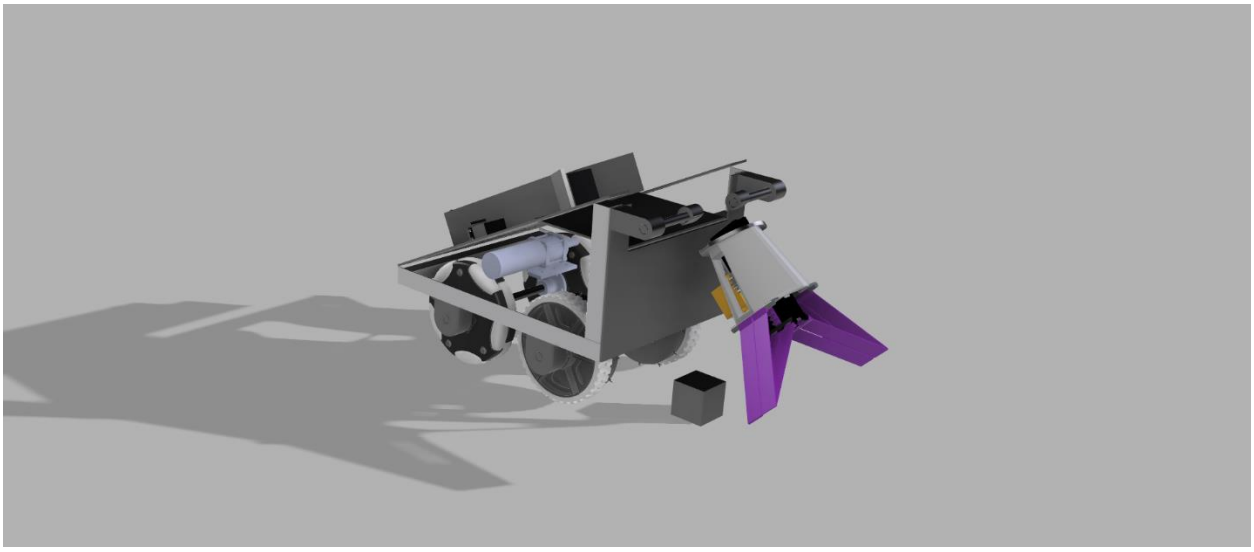
(Datasheet)

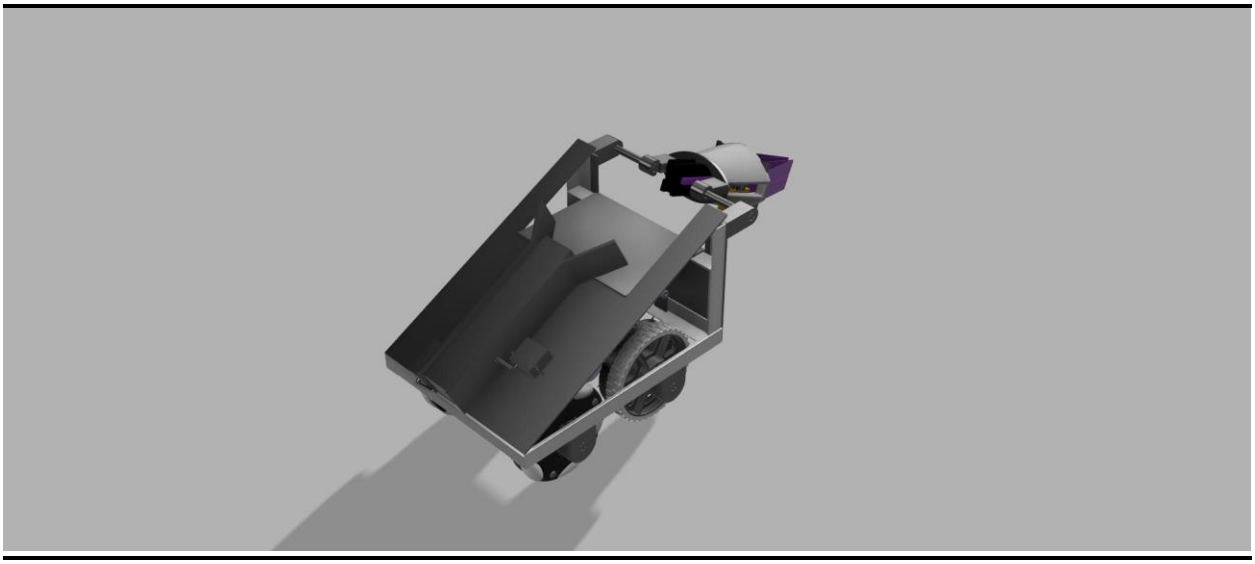
Appendix A: Arduino Mega 2560 Schematic

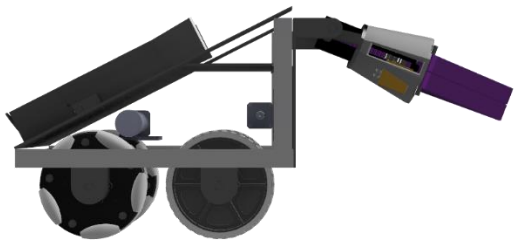


(arduino)

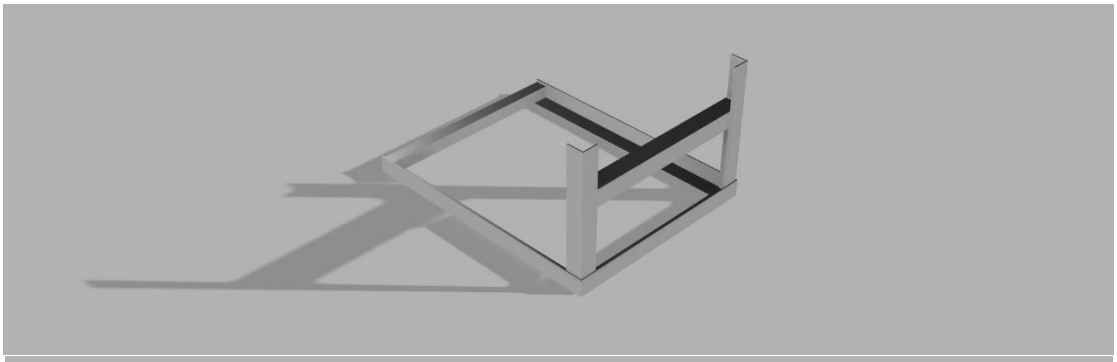
Appendix B: 3D Modeling

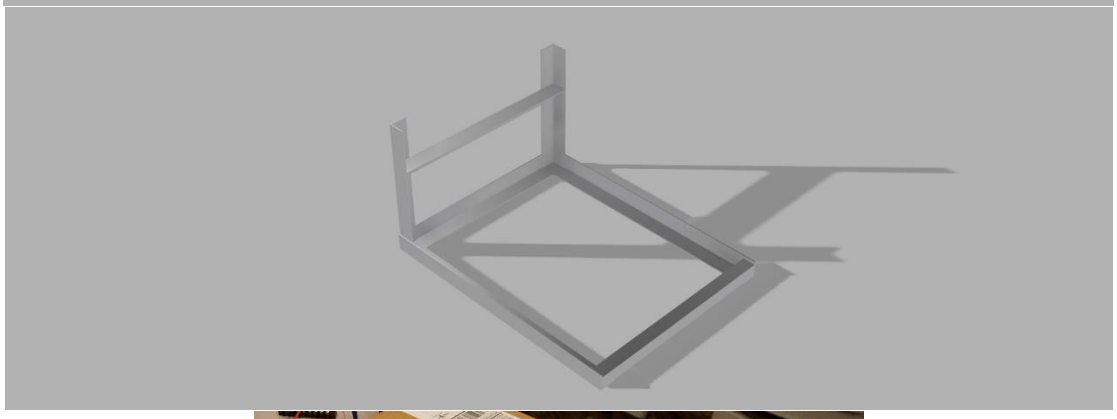


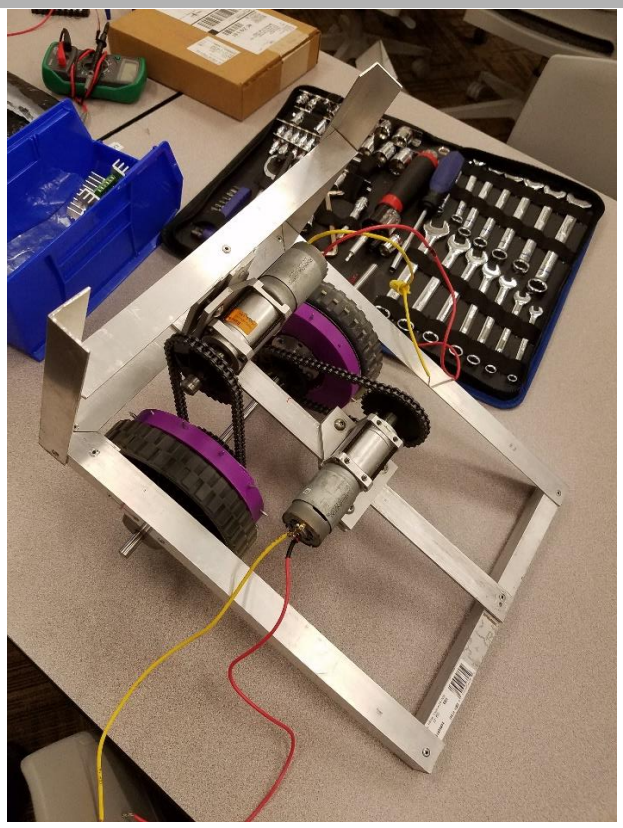




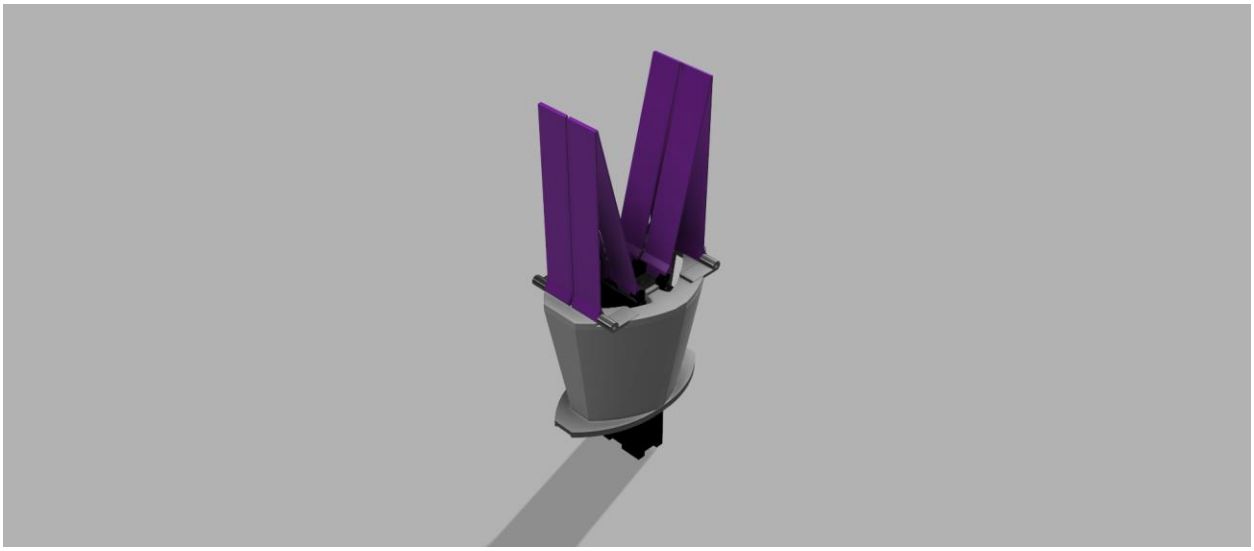
Appendix C: Robot Frame

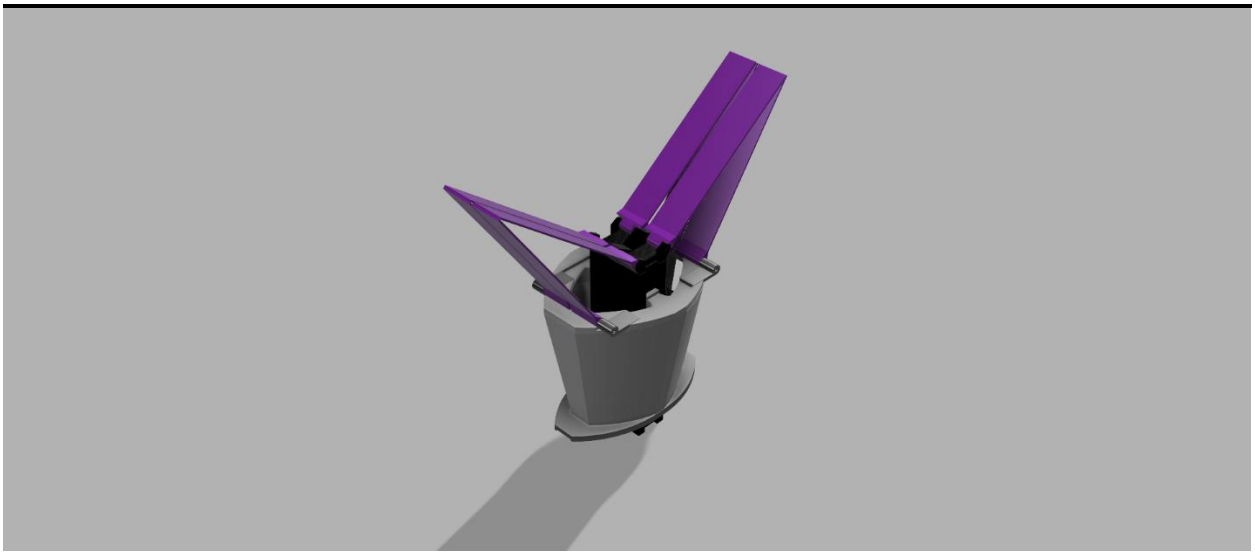






Appendix D: Gripper Arm





References

- Datasheet - 16V Modules (n.d.). In *Maxwell Technologies*. Retrieved October 15, 2012, from http://www.maxwell.com/products/ultracapacitors/docs/datasheet_16v_series_1009363.pd

- arduino-mega2560_R3-schematic (n.d.). In *Arduino*. Retrieved October 28, 2012, from http://arduino.cc/en/uploads/Main/arduino-mega2560_R3-schematic.pdf