2015 - 2016 RASC-AL EXPLORATION ROBO-OPS COMPETITION

Final Report

University at Buffalo, *The State University of New York*

Faculty Advisors:

Dr. Kevin Burke Dr. Jennifer Zirnheld

Graduate Students:

Kyle Thompson	Anvita Upadhyay	Ajeya Anand
Hemanth Manjunatha	Harshita Kapadia	Abhimav Siwach
Kugan Veluswamy	Sneha Thazhathethil	Parag Jayant Datar
Akeem Francis		Michael Singletary

Undergraduate Students:

Livio Forte III	Michael Bill	Mitchell Rathbun
Margaret Donnelly	Mark Grassi	Bryan Chansamut
Jia Yong Zeng	Akshay Gupta	Andy Neu
	George Melero	

University at Buffalo

The State University of New York

Contents

1 Introduction

The University at Buffalo Space Bulls Robo-Ops team has designed a new planetary rover, Astraeus III, for competition at the NASA Johnson Space Center. The rover is made of a single chassis with a unique active suspension. The suspension is controlled by four linear actuators, positioned above each of the four wheels, which adjust to keep the chassis level using an onboard inertial measurement unit (IMU). This design increases the stability by maintaining a level chassis while climbing steep slopes and traversing various terrain conditions. A linear actuator driven two segment arm with an end effector is used for sample retrieval with the ability to move the wrist while encasing the captured sample. A front mounted, rock collection basket is used for holding the samples. Four high definition webcams are used with pan and tilt capabilities for driver vision, blob detection and audio streaming. The electrical system includes a custom control board which allows for accurate and faster control of the linear actuators.

2 System Description

Capabilities of the Astraeus III are improved over last year's model, Astraeus II. The newly designed active suspension gives Astraeus III the ability to control ride height and weight distribution. Utilizing a new frame allows the rover to be better balanced and lightweight at 21 kg. The arm has also been redesigned, but follows the same operating principles as last year's model. It is able to reach and retrieve samples from a 0.4 m^2 area with ease.

Figure 1: 3D Rover Rendering

2.1 Drive System

2.1.1 Chassis

The chassis was assembled using aluminum plate and extrusions. To reduce weight, the electronic compartment's footprint was reduced and structural modeling was used to reduce wall thickness of material. To meet the desired weight, emphasis was placed on engineering for a factor of safety of 1.5 with a step force of 200 N at the wheels. From our midterm prototype, it was determined that the minimum amount of space required for the electronics was 46 cm x 46 cm x 15 cm. To make space for electronics, a two level layout was implemented in which the shelves are made out of high-density polyethylene. The final chassis overall dimensions are 53 cm x 46 cm x 22 cm and weighs 2.5 kg. The width and height of the chassis was also limited by the requirement that the wheels are square with each other when at ride height, for improved turning. To avoid impact damage when traversing obstacles, the suspension's linear actuators are isolated using rubber grommets.

2.1.2 Wheels and Suspension

The suspension is modeled as an active suspension, with a linear actuator controlling the four wheels independently. By controlling the extension of the linear actuator, the roll and pitch of the chassis can be manipulated to keep the chassis level. To achieve a target speed of 3 m/s, a brushless motor and planetary gear-reduction are being utilized. With the increase in speed, Astraeus III is more susceptible to vibrations. To aid in damping the shock vibrations, 17 cm foam core ($1/8^{th}$ scale monster truck) all-terrain tires will be implemented. The suspension length was determined from the chassis dimensions, so that at half extension of the linear actuator, the wheels would lay on a perfect circle. This is important, since the rover uses differential style steering, and if the wheels are not equidistant the wheels would be forced to slide across the ground. The height of the chassis at half extension is called the ride height, which is 19 cm off the ground, and is measured from the ground to the bottom of the chassis. The minimum and maximum height is 10 cm and 29 cm, respectively.

Figure 2: Astraeus III at Minimum and Maximum Suspension Height

2.2 Sample Retrieval System

The arm is designed using a series of revolute joints utilizing interlocking laser cut square aluminum tubing to increase the speed of rock retrieval. The arm weighs 2.9 lbs. due to the use of the laser cut tubing and interlocking joints. The joints are powered by a pair of 100 mm Firgelli linear actuators, one each to control the shoulder and elbow angles. The gearing option is 63:1 with a maximum speed of 20 mm/s and 100 N of lifting force. The reduced weight and higher gear ratio yields faster operation speeds. Furthermore, with its strength the arm can move small obstacles away from desired samples for easier retrieval. Testing is being done with two linear actuator pairs to further increase this pushing strength. The linear actuators are mounted within the aluminum tube section to prevent them from being exposed, hit or broken. The mounts also include soft 40A durometer rubber grommets. Testing has shown that an arm that is too rigid has the tendency to exert pressure on joints and break at the weakest point; the plastic actuator mounting holes. These grommets absorb some of the pressure put on the arm when pressing again object helping to prevent actuator damage. With improved arm control the sample retrieval system is even more responsive to the driver.

The manipulator uses a 3D printed two half semi-cylinder scoops. 3D printing has allowed for rapid prototyping and intricate shapes to be produced. The manipulator is designed to accommodate the varying rock sizes found in the rock yard. The design does not fully encapsulate rocks but clamps down on them using a high torque servo (611 oz-in, 4.3 Nm) and closing spring. Both halves close in together using two gear connected four bar linkage mechanisms. The system has four degrees of freedom including a tiltable end manipulator, with 0.69 meters of forward horizontal reach.

2.3 Vision System

This year a deployable mast is used for improved antenna radio reception and camera visibility. The mast is raised by a solenoid latch releasing a compressed spring. At a length of 0.72 m the mast can be at least 0.93 m off the ground, depending on suspension level. The camera placed on top is suited for long range vision and identification of areas of interest. The other drive cameras are mounted on pan-tilt (PT) servo mounts to increase their field of view. Operators will be able to control these cameras to view areas of interest. The mount is also isolated with rubber shock mounts to reduce vibration and improve image quality.

3 Control and Communication Systems

3.1 Control and Communication Systems Overview

Two computers, one on the rover and one at home base, are used in the communication system. Both computes run Ubuntu on an AMD quad-core processor.

The user interface is broken up into two parts this year, with those being a main window and a detection window. The main window is dedicated to displaying the information necessary for the rover to drive and pick up rocks. The top half of the screen is used to display the drive camera feed. The rest of the interface offers displays for various servos and actuator values from the rover, along with displays for the rover's sensor values. In addition, there are controls in the middle of the screen that allow for the user to start and stop the current camera, select the camera, change the Frames Per Second, and display the detection window, which will be explained later in

this paper. The UI thread is also responsible for receiving commands from the Xbox controllers and sending them to the rover computer through a TCP connection.

The detection window was added to the interface to help with the detection of rocks, as it was decided that more was needed than just blob detection running in the background. A separate stream is received for the two side detection cameras and presented in the top half of the window. These two cameras are mounted on servo controlled PT mounts, which allow for the cameras to have a wider range of view. In addition, this window allows for blob detection to be added to the video stream, and also presents a Google Maps API aided path tracking application that leverages the GPS coordinates obtained from the Rover's modem.

The rover computer has multiple threads running, with two major functionalities being achieved. The first was to listen for the commands being sent from the home computer and react to them. These commands included Xbox commands that dictated the rover's motion by specifying how the motors and arm actuators were to act, and camera commands that were used to select and configure the four cameras being used. The other was to send data back to the home computer, which was then usually displayed in the GUI. Overall, the rover computer acted as a type of server, with the client home computer sending commands to the rover and receiving information back.

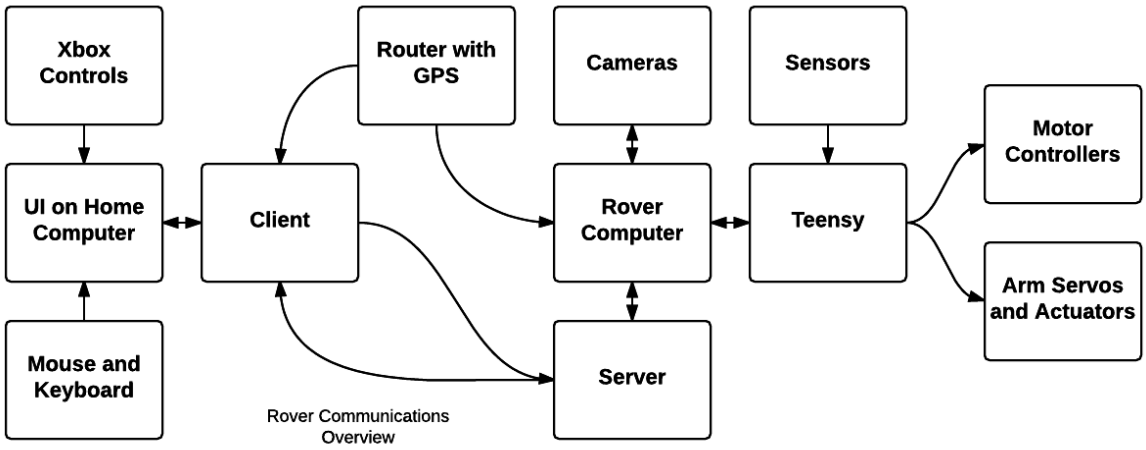

Figure 3: Communication System

3.2 Control System

The decision was made to use two Xbox 360 controllers, as this control scheme gives us a good amount of control over the rover and is fairly intuitive. One controller was dedicated to the drivetrain, while the other was dedicated to controlling the arm and picking up rocks. For the drive controller, each analog stick was mapped to the motors on the corresponding side. This allowed for tank style steering that gives the user full control of the rover's movement while still being simple and easy to pick up. The right trigger on the controller acts as an analog speed modifier, while the buttons are used to change the camera that the driver wants to see. Arm controls were focused on total control over the arm, which was achieved by mapping the movement of each servo and actuator to a button or button combination. As a result, the user is able to obtain the required accuracy and precision needed for quickly picking up rocks.

The various Xbox 360 controller values are picked up through a thread running concurrently with the UI. The PyGame libraries are leveraged to get these input values in a control loop, which interacts with our communication system to send the values to the rover computer.

The rover computer made use of ROS, or Robot Operating System, which is a framework that offers libraries and tools to simplify the process of controlling robotic systems. Doing so made communication between the various hardware we had to interface more efficient and modular, as code is broken down into separate nodes. A node in ROS is basically an executable file connected to ROS. Communicating between nodes is made simple by ROS through the use of topics. Each node has the ability to publish or subscribe to a given topic. We took full advantage of this pattern by running multiple nodes concurrently and relying on topics for communication between different areas of the code.

When a command string from the home computer reached the rover computer, it was published to a topic. A single node subscribed to this topic and parsed each string using python's built in string libraries. After determining the proper destination of the string, it was published to a different topic corresponding to its end usage. Each of these topics were in turn subscribed to by a node dedicated to a specific component of the rover's control system. Strings either represented a control command, a camera command, or a reset command. Below is a diagram of all the nodes and topics working together.

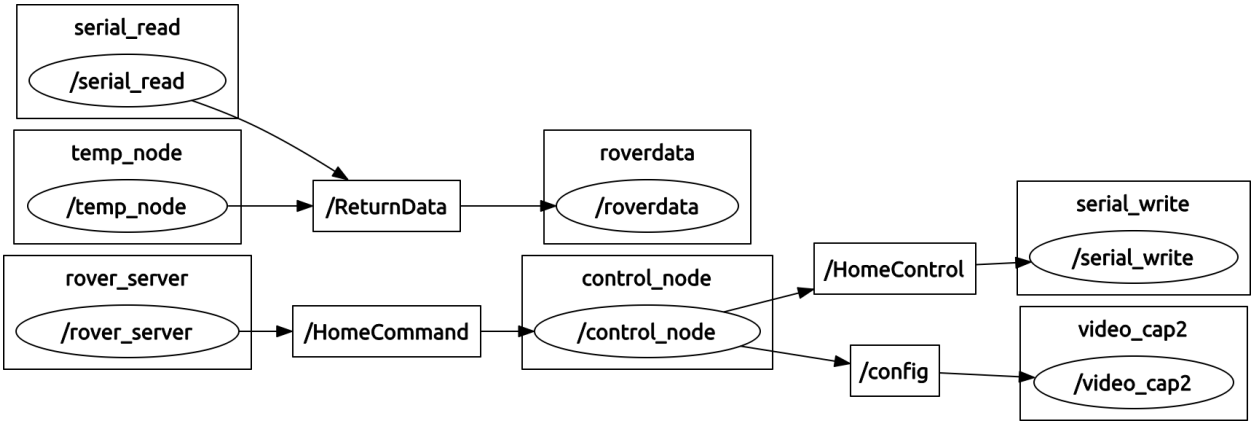

Figure 4: Communication Paths

The control commands simply pass a list of values to be written to the arm, motors, or PT servos. Due to the addition of custom actuator boards, the control scheme for the arm had to be slightly modified. Since the actuators are now speed controlled instead of position controlled, this was taken into account in the corresponding command strings. A node is dedicated to send the received string along to the onboard teensy. This is done with serial communication and the computer serial port. The teensy then parses the string and writes to the corresponding servos and actuators. The modularity of our control code allows for much easier debugging and problem detection, along with easy modification of movement speeds.

The camera commands consisted of four separate values, with one for the specific camera we wanted feedback from, and one for FPS, resolution, and screen width respectively. Since two streams were being sent at once, special caution was taken to make sure that two cameras on the same USB hub couldn't be selected at once, as this would cause an error to occur and the system to crash.

The reset command was particularly difficult to implement. The purpose of such a button was to remotely restart the computer without having to shut down all of the systems. Doing so would reset all connections and initial values, and served as a method to prevent a shaky connection or signal from ending a run. This was implemented by having a button on the UI set a flag. Once this flag was detected by the rover computer, all of the ROS nodes were killed. In our launch file, all of the nodes were made to be respawnable. This meant that once they were killed, each node would restart, allowing for the reset button to have its desired effect.

All of the controls were also implemented locally, which is both required for the competition and helpful for presenting the rover at various outreach events. To do this, it was decided that the ROS joy libraries would be implemented as opposed to the PyGame libraries used in the long distance control. This was because the PyGame libraries could not be used since the rover computer was essentially operating in 'headless' mode and PyGame is a gaming library. The ROS joy libraries provided a unique challenge, as a node in the joy library can get commands from multiple controllers, but can only write to one topic. Since the two Xbox controllers were needed for different areas of control, it was decided to run two joy nodes and map them to different topics. Controls were also added locally to be used as a sort of safety measure when demonstrating during community outreach events. The arm controller was made to be able to cut off any communication to the drivetrain, so that kids could try controlling the rover without there being any safety concerns.

3.3 Communications

Both UDP and TCP sockets are used for communication between the rover and home computer. UDP was chosen for video streaming because packet loss is not as important as limiting bandwidth and having a smooth resulting video. On the other hand, the guarantees provided by TCP are essential for the control system to function properly. If the rover receives the control sequences out of order the desired effect will not be achieved. By leveraging the benefits of both protocols, the communications system was able to limit bandwidth while still guaranteeing mission critical control.

As mentioned, control commands and sensor values are sent between the rover and the base computer through the use of TCP sockets. All of the data that must be transported is made into CSV strings to allow for easy parsing on the receiving side. The camera and drive information received at the rover is then published to the corresponding ROS topics to switch between cameras and move the rover. Feedback data is also constantly sent from the rover back to the home computer. The values sent include arm and actuator values, orientation data (roll, pitch, and yaw), and the CPU temperature. All of the data is displayed in the home UI with the CPU temperature being used as a sort of ping that allows the team to check the connection with the rover computer. This is done by simply concatenating the time to the end of temperature string that is being sent, thus displaying that time when it reaches the rover UI.

The rover computer connects to Verizon's 4G LTE network through the use of a Sierra Wireless AirLink GX450 modem. This modem was chosen due to its compact design, I/O flexibility, and compatibility with Verizon, which offers strong connectivity in and around Houston. The Sierra Wireless AirLink GX450 has a wide input voltage range and is powered from the rover's 12V regulator. It is connected directly to one of two Ethernet ports on the onboard computer and configured with a static IP address so as to give the client home computer an access point.

3.4 User Interface

The UI was implemented using the QtDesigner tool. This was done for the team to separate the UI's look from the code responsible for its logic, allowing for a more flexible system. While the visual design was fairly simple, the UI was responsible for multiple tasks. The most important feature was displaying the rover camera view on our base computer. This window was chosen to be fairly large, as it gave us a better view of the rover's surroundings. It also helped with the precision needed when picking up rocks. The UI also displayed the values being written to the rover's control system, which allowed for the user to have a better idea of the rover's positioning when dealing with delay and/or a poor camera feed. Furthermore, values were read back from the rover, such as the temperature, which was displayed alongside the time that the string was sent.

The Xbox controllers were integrated with the UI to control the rover effectively. For this to occur successfully, our TCP communication code also had to be integrated so that we could seamlessly send commands to the rover. Command strings were also sent to select cameras along with given characteristics of the camera stream.

A new addition this year is the use of a second window to help with rock detection. This window has two parts. The top half of the window displays the current side camera. As with the main window, this display is chosen to be fairly large to give the user the best view of the surrounding environment. Both the main window and the detection window are responsible for listening to keyboard events, which are parsed and then used to control the pan, tilt, and zoom of the onboard cameras. The bottom half of the window is dedicated to displaying information obtained from the modem's GPS sensor. The Google Maps API is leveraged to compare this data against a map of the surrounding area, allowing for the team to avoid potential danger areas and estimate the relative positions of various rocks. Blob detection was made to be a secondary option this year, as something that can be added to the received detection camera streams.

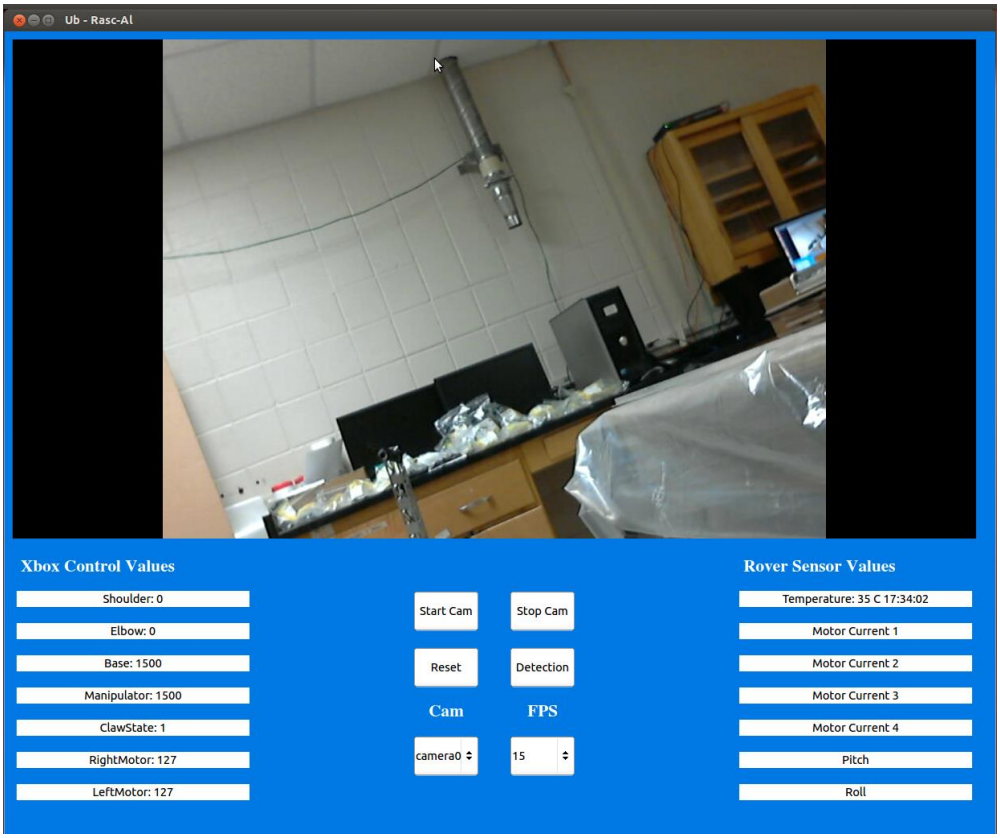

Figure 5: User Interface

3.5 Camera and Vision System

Four Logitech C930e cameras are being used this year for the team's vision system. These cameras were selected due to their excellent image quality, size and ease of mounting. In addition, the cameras offer digital zoom and a wider range of view than the team's previously used webcams. The Logitech C930es also have built in microphones, which as testing has shown, work well for recording audio of the rover for the competition. The front drive camera microphone will be used to record audio for the live stream.

The four cameras are mounted randomly with integer IDs 0-3 when the rover starts. However, we need to know which ID corresponds to which camera. To solve this problem, the team leveraged Linux's udev rules to establish soft links to the camera ports. All of the C930e USB webcams are identified by their serial numbers. The cameras are then remapped with integer IDs from 4 to 7. By doing this, each camera can be selected by the same ID each time, regardless of how the computer initializes them.

The GStreamer Library was used to handle the video streaming this year. GStreamer is an open source multimedia framework that allows multiple multimedia elements to be piped together. MJPEG encoding was decided on after extensive testing, as it gave the best final image. While MJPEG does result in a higher bandwidth than encoding such as H.264, the Real-Time Transport Protocol was used to limit these effects. With commands from the home computer, the GStreamer commands were also easily made to switch between fps and resolution settings.

Blob detection has been made an optional addition to the received video stream in the detection window, as the two detection cameras are no longer dedicated blob cameras and can be streamed with or without the image processing added. To successfully implement blob detection,

the OpenCV library is used to analyze the images in both streams. First, the camera frames are converted to HSV color space. Hue, saturation and value ranges for each color are then defined. If a set of pixels in the image all belong within a given range, a blob is formed around them. Each blob is represented by a bounding box which are color coded in order to be easy to distinguish visually.

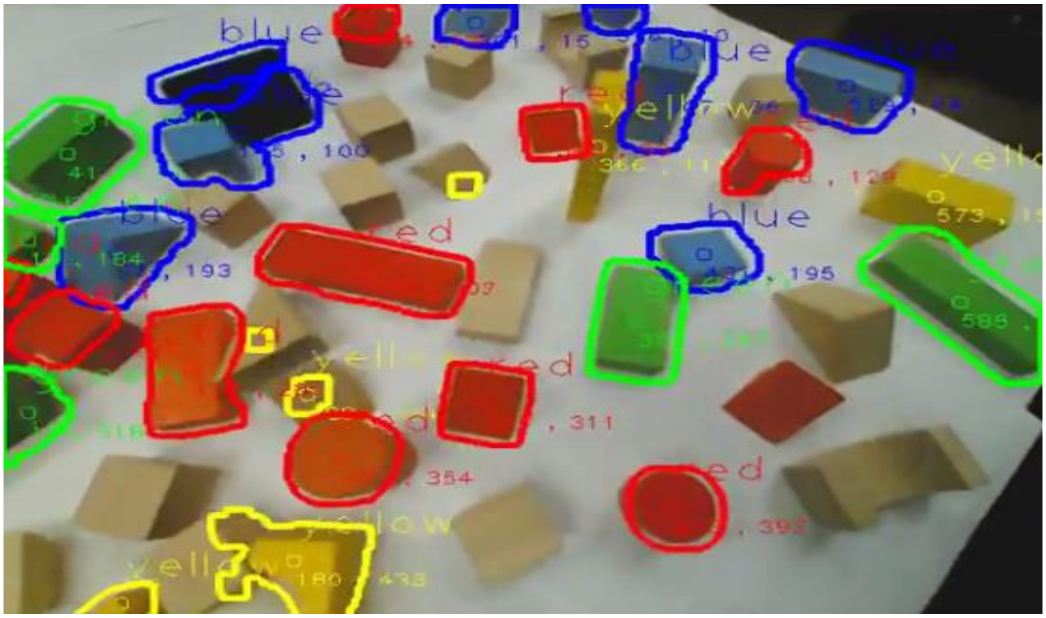

Figure 6: Blob Detection

3.6 Hardware Controls

The control system is driven by a Teensy 3.2 microcontroller which is interfaced with the computer through a serial connection. The choice of controller provides ample room for processing sensors and controls due to the high speed of the controller (96 MHz). Communication with the Teensy occurs in a packet setup wherein the computer sends several groups of comma separated values which send set the speed for the drive motors as well as the positions for all the actuators and servos.

All of the hardware interfaces with a custom designed board that allows for reliable connections between the microcontroller and the hardware. The schematic for the board is shown in figure 7 along with the Printed Circuit Board (PCB) layout in figure 8. The disadvantage to the Teensy is the limited number of I/O pins. To extend this a MCP23017 16 bit I/O expander is used. This gives 16 extra digital outputs, with the cost of only two I2C pins. To extend the number of analog inputs, the 8 input 4501 analog multiplexer is used, which costs 3 pins. These two together add an additional 19 pins to the Teensy to be utilized by hardware onboard.

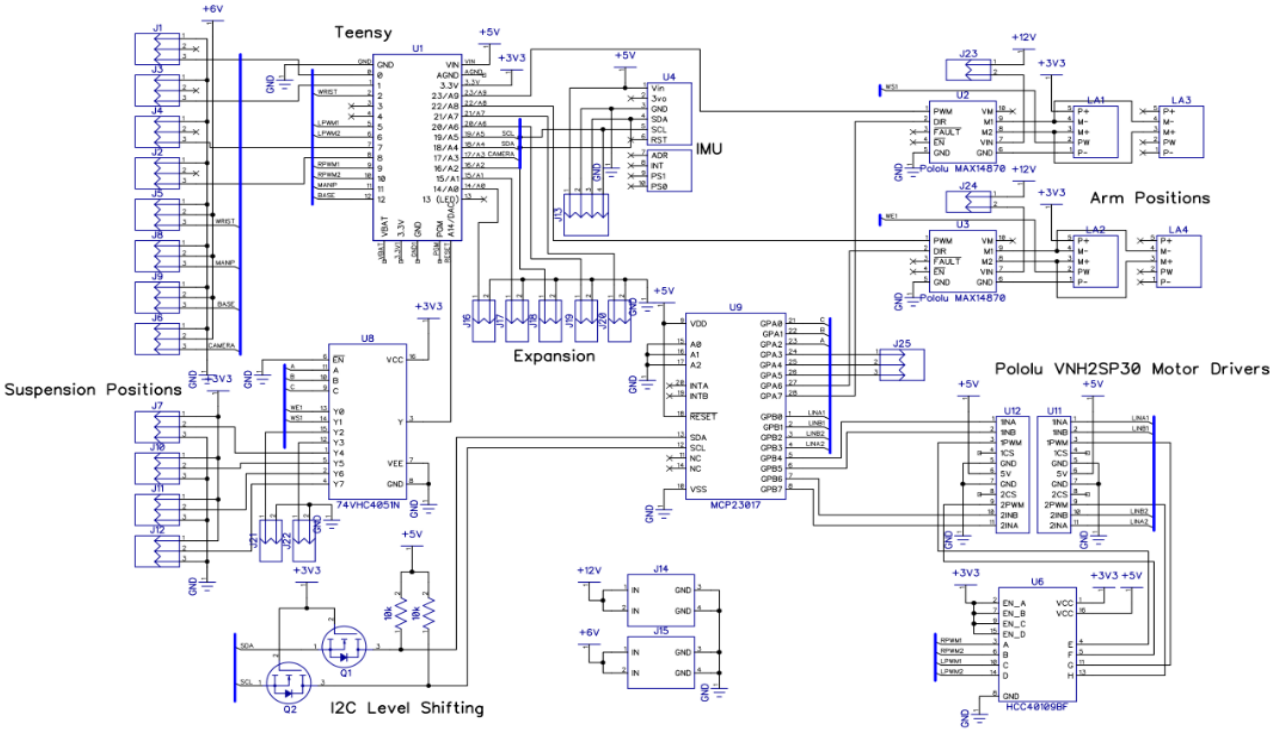

Figure 7: Custom PCB Schematic

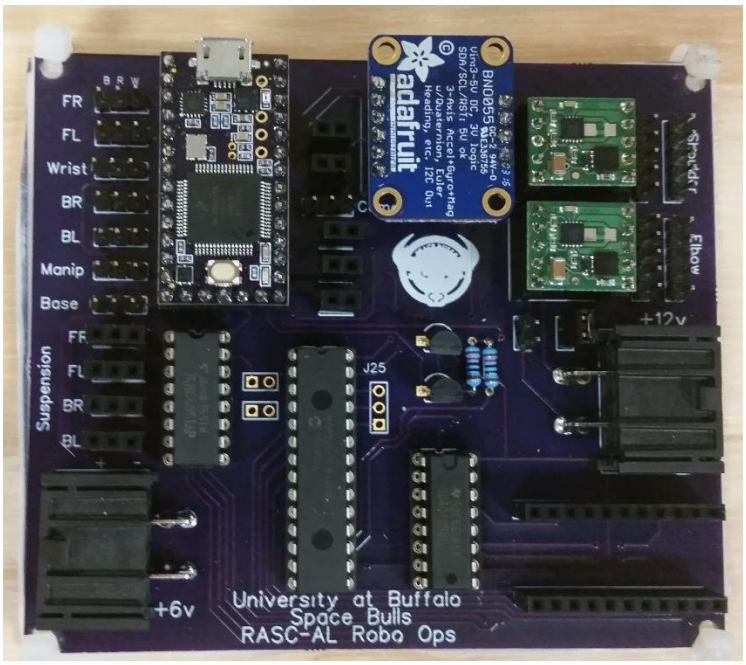

Figure 8: Custom PCB

The linear actuator controls for the arm have been greatly improved in terms of speed and usability. The arm actuators typically use positional control boards, which are slow and make intuitive control difficult. Small DC motor controllers, the Pololu MAX14870, are used for direct control from the microcontroller. This allows for control of the actuator speed which makes for very intuitive controls. The actuators contain a feedback potentiometer, which can be monitored for positional control in situations such as automating positioning of the arm before or after sample collection.

The actuators for the suspension use a similar DC motor controller, but designed for higher power motors, the Pololu VNH2SP30. Positional speed control can both be achieved by the suspension actuators. To monitor the pitch and roll of the chassis, the BNO055 Inertial Measurement Unit (IMU) is used. This board combines an internal gyroscope, accelerometer, and magnetometer data to provide extremely accurate and stable angle measurements. Proportional control of the pitch and roll is used to calculate adjustments for each axis individually. These values are then linearly combined to achieve two dimensional correction of the chassis.

To accommodate the new brushless drive motors which demand high inrush currents, a new battery had to be chosen. A Multistar four cell lithium polymer battery was chosen. It has 20000mAh capacity so run times of over an hour can be achieved. It has a 10C discharge which allows for ample current to drive all the motors and the rest of the electrical system.

To control the drive motors, HobbyKing 60A Brushless Car Electronic Speed Controllers (ESC) are used. These are controlled just like servos, allowing use of the standard Arduino Servo library, which uses pulse width modulation (PWM) by sending small pulse widths to control the speed. The motor controllers are directly powered from the battery.

4 Technical Specifications

Table 1: Technical Specifications

Weight/ Rated Payload	21 kg
Max Speed	4 m/s
Drive Motors	736 W 1980 KV Brushless Motors, 69:1 BaneBots Gearbox (x4)
Suspension Actuators	5 cm stroke, 155N Dynamic Force (x4)
Max Obstacle Height	26 cm
Max Obstacle Width	35 cm
Suspension Travel	18 cm
Battery	14.8 V Lithium Polymer 20000 mAh 10C
Communications	Sierra Wireless GX450 Mobile Gateway
Computer/software	AMD A10-6700, Ubuntu

5 Mission Control Center Operational Plan

Mission control center will have three primary drivers. One driver will be controlling the rover drive and suspension, while another driver controls the arm and manipulator for sample retrieval. The third and final driver will be in control of the pan tilt cameras for sample and obstacle detection. This person is going to watch the blob detection cameras for samples and will also be the primary decision maker stating where to drive and what rocks to collect. This will ensure sufficient time to traverse all terrains and return back to the start for bonus points. Mission control will have two monitors for the driver's camera streams with a plan of attack, timer and map projected on screen at the front of the room. The primary decision maker is the camera driver and has the final say in conflicted decisions. Backup computers and controllers will be on hand and ready to swap in event of a problem. Other team members who have training for each system will be standing by in the event that they are needed, when not driving they will be looking for rocks on the monitors. Simulated timed runs also helps to develop team cohesion and check functionality.

6 Testing and Overall Strategy

Testing is being carried out on terrain similar to that seen in the gravel crater, sand pits and Mars hill. Results have shown that the new tires have improved traction on all terrain surfaces and reduce vibration transmitted to the chassis compared to rigid wheels. The new tires also prevent sand and gravel accumulation in the rims.

The active suspension is being tested on various terrain types to check functionality. The control algorithm is constantly being improved for greater stability. Testing also included roll over and tipping tests during which the rover performed in an acceptable manner. The design is also modular, which allows for easy modifications.

6 Budget

Table 2 - Competition Expenses

Competition Expenses		
Vendor	**Item**	**Cost**
NIA/NASA	(4) Competition Registration Fees	$800.00
Enterprise	(7 day) Rental Van	$700.00
Hilton Houston	Hotel Rooms	$1,144.92
Various	Gas and Tolls	$500.00
Total		***$3,144.92***

Table 3 - Material and Equipment Expenses

Material and Equipment Expenses		
Vendor	**Item**	**Cost**
Adafruit	Sensors, PTZ Camera Mounts	$323.92
Amazon	Computer, Electronics	$1,920.76
BaneBots	Gearboxes	$792.55
CMS magnetics	Camera Mast Magnets	$11.92
Digikey	Electronic Components	$166.70
Firgelli	Arm Linear Actuators	$360.00
Firgelli Automations	Suspension Linear Actuators	$783.94
HobbyKing	Motors, ESCs, Batteries	$582.21
McMaster-Carr	Hardware	$196.26
Merritt Machinery	Machine work and material	$550.00
Newegg	Bearings	$16.98
Osh Park	Printed Circuit Boards, Teensys	$138.20
Pololu	Motor Controllers for Actuators	$373.54
Pro-Line Racing	Tires	$230.99
RobotShop	Servo Mounts	$38.09
TineyShine	Servo Mounts	$25.90
Total		***$6,511.96***

Table 4 - Sponsorship and Donations

Sponsorship and Donations		
Vendor	**Item**	**Cost**
Firgelli	Discount on all purchases	$320.00
Firgelli Automations	Discount on all purchases	$335.92
Pololu	Discount on all purchases	$96.17
Merritt Machinery	Machine work and material	$1,500.00
Custom Laser	Laser cutting	$1,500.00
Sierra Wireless	(2) 4G Modems	$1,398.00
Verizon Wireless	4G LTE Service Plan	$2,000.00
UB Computing Services	Dedicated Wireless Access Point	$1,300.00
UB Science and Engineering Note Services	R&D Computers	$3,000.00
UB School of Engineering and Applied Sciences	Development Space	Priceless
Total		***$7,150.09***

This budget represents expenses for a completed rover, with a duplicate of all components, including the chassis. The end goal is to continue to do outreach with these rovers to peak interest in STEM fields for younger students (K-12 and undergraduate college students).

7 Public/Stakeholder Engagement

This year, the Space Bulls team has continued use of a Facebook page to share updates and information to our followers throughout our progress. This avenue allowed the public to get updates from us as they became available. The page has also generated many new views helping spread UB Space Bulls through the community

The Buffalo branch of the American Association of University Women (AAUW) hosted its eleventh annual Tech Savvy conference this past March. Being a national organization that advocate for equality and education for women and girls, this gave us a great audience for a broader impact to the lives of many middle school girls to pursue careers in STEM. This year the team hosted a “Rover Races” workshop, which is a NASA classroom activity.

To show our gratitude to the engineering school, the team combined with the UB IEEE Student Branch for open house events throughout the semester. This allowed us to reach current and prospective students and parents for the ongoing work. We used this to showcase a multidisciplinary project that included undergraduate research and collaboration with graduate students.

The team also attended the Buffalo Museum of Science as part of their Energy Day. Here we gave young children and adults the opportunity to drive last year’s rover and talk about the competition and foster interest in engineering

In May, a few Space Bulls representatives joined the UB President, Community and Donors at Party On Point to celebrate student achievement made possible by philanthropy to UB.

Students spoke with the attendees regarding the project and the great opportunity it has provided them personally and professionally.

8 Moving Forward

Moving forward over the next two weeks the team prepares for further extensive testing and driver practice in simulated, competition-like scenarios. The rover will be tested for endurance and maneuvers for problematic situations planned out, for example stuck wheel. The rover will be fully disassembled for metal to be powder coated and final finish to be applied.

9 Acknowledgements

The UB Space Bulls team would like to sincerely thank all of our donors, sponsors, and supporters. Through the support of the School of Engineering, our advisors and faculty members we were able to see this opportunity through to the end. Without the generosity of our peers, donors and industry partners we would not be as successful as we are today.