RHEA-1

Remote Harvester, Earth Analog 1

University of Maryland, College Park

Faculty Advisor; Dave Akin
Student members: Ruben Thomas Abraham, Oscar Alvarado, Andrei Arevalo, John Bowen, Chris Carlsen, Feng Chen, Steven Friedman, Shimon Gewirtz, Shean Grahm Howell, Kyle Jordan, Matt Marcus, Jacob Moschler, Corey Simmons, Jonathan Sternberg, Gary Sullivan, Brennan Thews, Lauren Tullis, Matt Westerfield, Jingyou Xu

RHEA-1

University of Maryland, College Park

Abstract

RHEA-1 is a remotely operated vehicle, designed and built by an undergraduate class at the University of Maryland, College Park. It has been developed during the spring semester of 2011 to compete in the first annual Robo-ops competition. Rhea was designed with a strong emphasis on simplicity and reliability, using well known chassis and manipulator designs, as well as constructing the electronics entirely from commercial, off the shelf, components. In addition to constructing a tele-operated rover, Rhea's team has participated in multiple public outreach events, giving talks to local high schools and to the general public. Our focus is not only creating a functioning rover, but to make an educational tool to further the public's understanding of robotics.

Chassis

The chassis is based off of a welded rectangular platform, providing a simple mounting space for the rover's components. The battery is mounted below the frame in a padded aluminum shell, to prevent accidental damage the battery due to protruding obstacles. This provides a low center of gravity, reducing the risk of roll. On top of the battery, the electrical box is mounted, containing the computer, motor controllers, power converters, and power distribution block. In front of the electrical box the arm is mounted with baskets to store rocks on its sides. A suspension system, consisting of two A-frame rockers, allows for constant contact with all four wheels. The system is stabilized by four high tension springs, limiting the tilt of the chassis while traversing rough terrain.

Drive System

To move RHEA-1 we decided to go with a simple four wheel drive design. A gear motor is mounted at each corner of the rectangular frame. The motors are mounted directly to the chassis and skid steering is used to turn the vehicle in place, via differential turning. Each motor is connected to its own controller, which uses a rotary encoder to independently regulate the velocities of each wheel at a firmware level. The output shaft from the gearbox is directly connected to the hub of the wheel, reducing weight and play in the drive system.

Wheels

RHEA uses a porous polyurethane tires, produced by Care-Free Tires. These tires are puncture proof and lighter than traditional pneumatic tires. They have been custom milled with a high traction, square knob pattern to provide adequate draw bar pull. When fitted to a light weight, two piece, aluminum wheel, the final outer diameter is twenty centimeters. Testing has proved our drive system to be extremely capable, scaling vertical obstacles over twenty centimeters in height.

Arm and End-effecter

The arm was determined to require three degrees of freedom in order to function properly. The arm was designed as an assembly with two servo motor gear boxes facing each other mounted on a plate fastened to a third servo motor gear box. The arms were made from square tubes due to the flat surfaces for mounting. A piece of tube is attached to the hubs on one of the gearboxes and a smaller piece is attached to hubs on the gear box. A second piece is attached is on a rod that also goes through the first piece and acts as a pivot point. The small tube piece is attached to a hollow round tube with rod end bearings on either side. The second rod end bearing is affixed away from the pivoting rod on the square tube. The assembly of the arm can be seen in the CAD model pictured to the right.

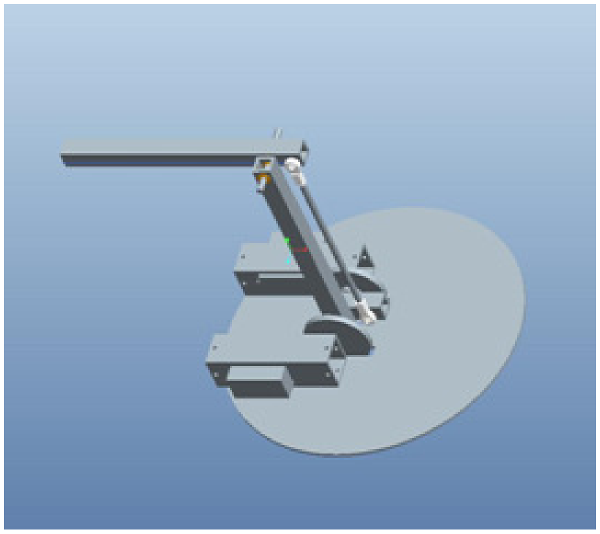


Figure 1 Cad model of the arm

The closer servo controls the first arm section, while the farther one controls the small rod connected to the second arm section. The third degree of freedom for the arm is due to the unseen servo beneath the plate, which gives it 360 degrees of motion.

The end-effecter is made with a four finger design, each finger is made from aluminum sheet bent to a c-channel to reduce weight and add rigidity. A linear actuator is used to open and close the claw. It is bolted to the top of one aluminum block and the threaded rod passes through the block and is bolted to a second aluminum block. The bottom aluminum block is fastened between four brackets. The bracket assembly has holes at four points. The top block has four rod end bearings bolted to the sides. The rod end bearings are connected to a second rod end bearing by a threaded rod. The second bearing is connected to a bolt in the finger.

Each finger is thin on the bottom and thick on the top. The top contains two bolt rods in it. The top rod is connected to one of the holes in the bracket assembly. The bottom bolt is connected to the rod end bearing assembly that is connected to the top aluminum block. As the distance between the two blocks changes, the fingers pivot about the top bolt, closing the claw to hold objects.

The claw is connected to the arm through a pivot assembly. A fourth, much smaller, servo controls an arm that is connected to a round tube, similar to the one with the rod end bearings on the arm, which is connected to the claw on the other end. The servo controls and stabilizes the angle of the claw as it is picking up objects.

Electronic Housing

The computer and all electronic controllers are housed inside of an aluminum sheet metal box. The components are mounted on brackets bolted through the box. Each hole is sealed and the sides and openings for wires are coated with weather stripping create a closed environment inside. The lid of the box is removable and has awnings on its sides to wick water away from the openings in the box. Exhaust fans are fixed to the box to provide negative pressure to properly cool the components and keep them within operating range. HEPA grade filters keep the dust out at the intake holes on the opposite side of the box.

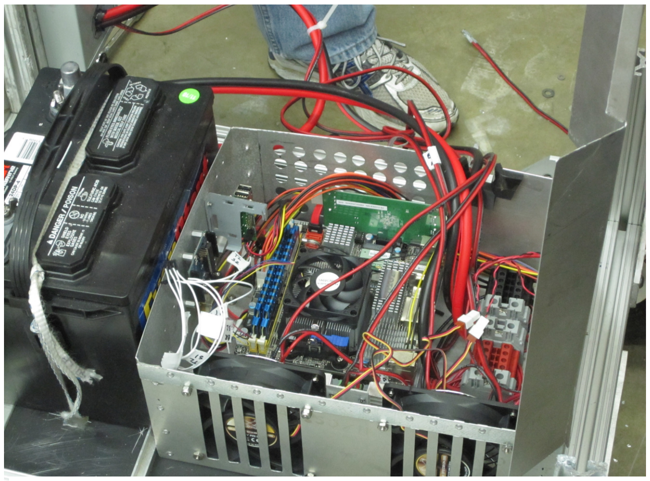


Figure 2 Electronics box interior, prototype chassis

Deployable Boom

In order to provide adequate vision of the surroundings, a boom is affixed to the rover on a pivot point. To stay within the stowage range the boom is initially held in the down position, parallel to the ground. A spring is fixed to the chassis and the boom, initially in the compressed position. When the actuator is moved, the spring releases, causing a rotation the pivot point, deploying the boom to its upright position. Cameras are affixed to the top of the boom, to provide vision for the driver to receive through a live stream.

Electronics

Onboard computing is done by a 3.1 GHz quad core AMD Athlon II X4 on a mini-ATX motherboard. The computer will run the top level programming for controls, but is also used for real time video compression. The high clock speed and multiple cores of the Athlon processor enable low latency compression and transmission from the rover, giving the remote operator’s station a real time view of the competition field. This light weight computer system has been tested under extreme vibration and has proven to be very reliable.

Rhea uses one Arduino Uno and one Arduino Mega, supporting ATMega chips. The Arduino Uno, handles communication with the computer and controls both the wheel motors and arm motors. The Uno is connected to the computer through a USB cable and receives information with identifying byte followed by the raw data to directly specify the speed or position of the motors. The computer sends raw data in order to keep as much processing as possible on the computer and away from the interfacing hardware. The Uno also has a one second watchdog timer which checks for a signal from the computer and turns the motors off if one is not received in order to prevent the Rhea from running away if the computer malfunctions. All communication is handled by the open source serial communication library on the Arduino. In addition, the wheels are controlled through servo library which sends a pulse signal with varying widths which correspond to different speeds and directions. The servo motors in the arm are controlled by the same servo library, where the pulse widths correspond to different positions on the servo.

The Arduino Mega's only purpose is to count the amount of wheel rotation from each rotary encoder and send that raw data to the computer. These rotary encoders work by outputting two offset sin waves. Each rotary encoder is set up to an external interrupt that triggers every time the signal changes, which indicates a preset angle of rotation of the wheel and can also provide the direction the wheel is turning. Both Arduinos communicate to the computer through drivers written in C code, where all information is sent as either a one or two byte unsigned integer.

Arduino code was tested through software provided by the Arduino Company which handles serial communication and by providing manual inputs to the system and checking the output. In addition, during mechanical or software testing of the vehicle problems that could be easily fixed through firmware were accounted for.

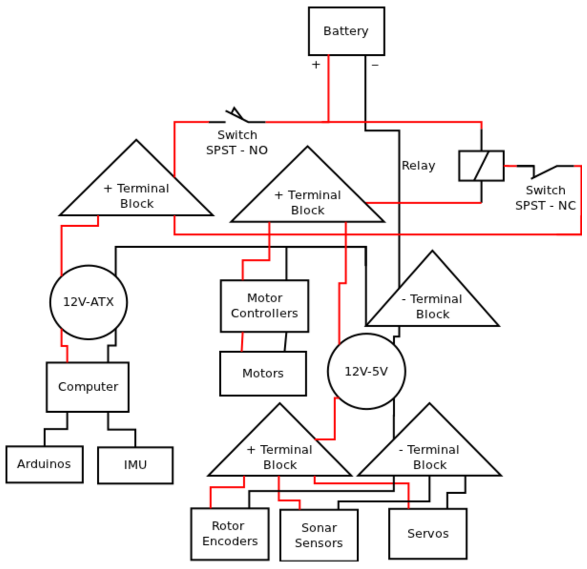
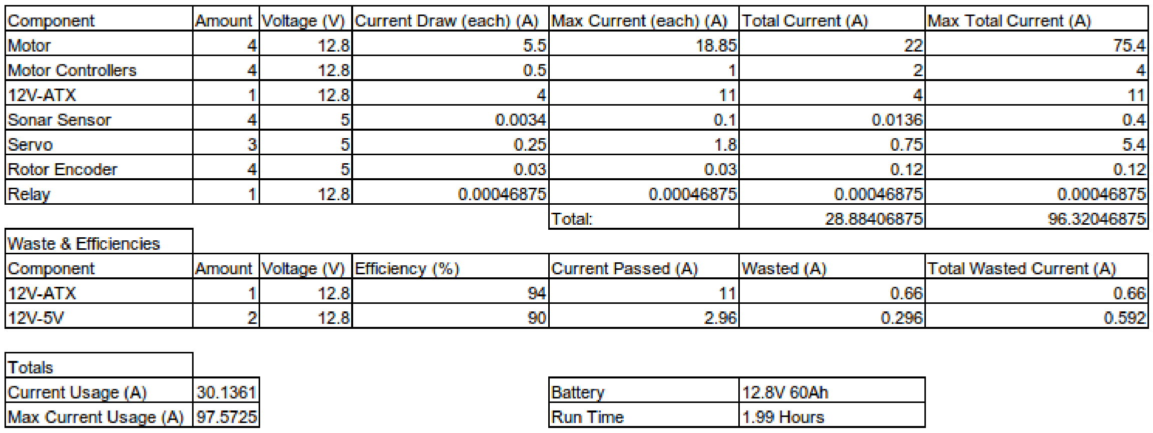


Figure 3 Power distribution chart

The battery that we are going to use is a 12.8V lithium iron phosphate battery that has approximately 60 Ah and can output 100 amps continuous, 150 amps peak. It also has a built in balancing PCM board, which is a help to keep all of the cells charging and draining continuously. We expect that this battery will last approximately one and a half hours, based on experimental power draws. The power system was overdesigned to permit a higher continuous draw from the motors. This will account for the additional power draw, peaking at a net of 90 amps, used when traversing obstacles.

The power distribution chart is shown in Figure 3. This shows our use of terminal blocks and converters to distribute the power between the components. We are using DIN Rail terminal blocks and have a common ground and two separate positive rails. One of the positive rails has a 30A switch and it controls power to the computer. This ensures that the leads will not spark when attaching the battery and that we can keep the computer off even when the battery is



Component	Amount	Voltage (V)	Current Draw (each) (A)	Max Current (each) (A)	Total Current (A)	Max Total Current (A)
Motor	4	12.8	5.5	18.85	22	75.4
Motor Controllers	4	12.8	0.5	1	2	4
12V-ATX	1	12.8	4	11	4	11
Sonar Sensor	4	5	0.0034	0.1	0.0136	0.4
Servo	3	5	0.25	1.8	0.75	5.4
Rotor Encoder	4	5	0.03	0.03	0.12	0.12
Relay	1	12.8	0.00046875	0.00046875	0.00046875	0.00046875
				Total:	28.88406875	96.32046875

Waste & Efficiencies						
Component	Amount	Voltage (V)	Efficiency (%)	Current Passed (A)	Wasted (A)	Total Wasted Current (A)
12V-ATX	1	12.8	94	11	0.66	0.66
12V-5V	2	12.8	90	2.96	0.296	0.592

Totals	
Current Usage (A)	30.1361
Max Current Usage (A)	97.5725

Battery	12.8V 60Ah
Run Time	1.99 Hours

Figure 4 Power draw chart

connected. The other positive rail connects to both of the 12-5V converters for the servos and rotor encoders as well as all four of the motor controllers. This is connected to the battery through a 200A relay with a kill switch. This setup is ideal for testing as it allows us to kill power only to the wheels and the arm even when the computer is running in order to preserve logs in case something goes awry and when testing so the computer does not have to restart every time a test needs to be conducted.

The motor controllers we are using are SyRen 25. These are single motor controllers, which allow us to control each motor individually. We are using RC control with them since the Arduino has built in drivers to cover RC and that makes controlling the motors fairly simple. To test these we used both Analog and RC input, for the Analog input however we needed to use capacitors to filter the signal so RC was the safer route. However, when we send the same information to the motors, they do not all turn at the same rate, so the encoders will have to compensate for the discrepancy between them through a feedback system.

We are using capacitive, incremental rotary encoders made by CUI (Model- CUI AMT102-V KIT) as they are rugged and dust/dirt protected. The encoder has 16 selectable resolutions and we chose 512 PPR for maximum accuracy. Other benefits of the encoder include low current consumption (< 10ma) and a broad operating temperature (-40°C to 100°C). For testing these encoders, we will have a person with a stopwatch measuring time and number of revolutions and compare them to the encoder data that is received. This stage of testing has not occurred yet as our concerns were elsewhere, but is the next step on our list.

Rhea contains a nine degrees of freedom Razor IMU which uses an ATMega328, LY530AL (single-axis gyro), LPR530AL (dual-axis gyro), ADXL345 (triple-axis accelerometer), and HMC5843 (triple-axis. This provides triple axis rotational orientation and acceleration and a triple axis magnetometer to help the software controller and the vehicle's driver navigate. The IMU sends information to the computer through a USB.

We are using two different types of servos for the robotic arm of our rover. We are using a SPG805A with a gear ratio of 5:1 for the platform. It rotates the arm around the base and has a maximum rotation of 400°. The Servo can deliver a maximum load of 1375 ounce-inch at 5V input. For moving the smaller links of the arm we are using an SPG785A-A at a gear ratio of 8:6:1. It has a maximum rotation of 146 ° and can deliver a maximum torque of 1315 ounce-inch at 5V. These servos have been tested to determine the code needed to work, however the arm has not been completed to the point of an actual demonstration.

Software

Rhea's software was designed with remote operation in mind. Rhea takes many concepts from distributed networks for efficient communication between computers. Rhea's distributed nature allows for multiple computers to communicate with different portions of the robot concurrently. This allows multiple operators to handle different portions of the robot. For example, one person could control the movement of the robot while another handles the arm control.

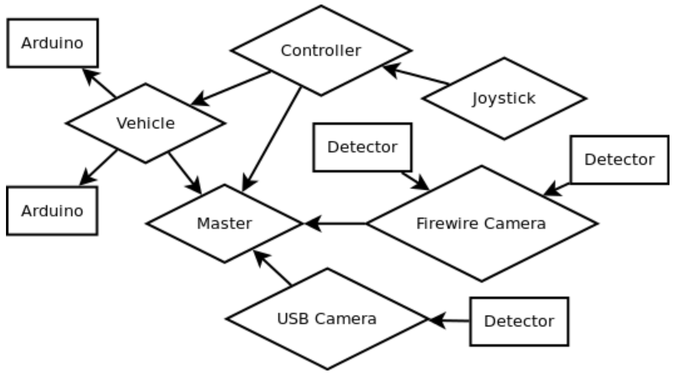


Figure 5 Control Diagram

The distributed platform that Rhea is built on allows for multiple languages to interface with the robot. These offer no performance cost except for the cost of marshalling/unmarshalling the arguments. For example, the main software is written in C++ on Rhea. The joystick code is written using Python because it was easy to create, modify, and reliable. This feature allowed us to create a Java Applet to do web streaming using the same interface that we use for our own streaming.

There are four important high level interfaces to Rhea.

- Master
- Vehicle
- Controller
- Camera

The Master interface provides a single entry point for finding the other interfaces. The Master is the first interface created during initialization and all of the interfaces register themselves with the Master node upon creation. The Master node is the only one kept with a consistent name and port. This allows an outside program, such as the video web server or the OCI, to find and communicate with an arbitrary subsystem. This interface is proved solely to make finding nodes simpler.

The Vehicle interface is a higher level abstraction over the hardware. It interfaces with the firmware to control the motors and the arm. It also runs a loop to continuously poll the motor encoders and IMU data. The firmware for the motors requires a continuous heartbeat sent at least once a second. The loop sends this heartbeat. This heartbeat is to tell the motors (which are controlled through Arduino's) that the computer is still alive and the software is still connected. It does not say whether we have lost our connection to Rhea itself. For this reason, the Vehicle implements its own keep alive signal. The joystick (which is on the client side) continuously sends a keep alive signal. The joystick is the only piece of code that sends a keep alive signal and so the robot can be stopped within a second of killing the joystick (on the software level).

The Controller interface is the primary interface for telling Rhea how to move. The Controller gets the motor encoder data from the Vehicle interface and tells the Vehicle how to move each wheel to reach a specific speed. There are two Controller interfaces on Rhea: the wheels and the arm. A PID controller is used for each wheel.

The Camera interface is a high level interface for a camera. There are two implementations of this interface. The first implementation accesses a physical camera using OpenCV. OpenCV was chosen because the video interface was able to communicate with both USB and Firewire enabled devices. There is a common problem with interfacing with physical cameras when a long running operation may prevent a new image from being polled. Camera's typically keep an internal buffer of captured images. This is so that frames are not lost unintentionally when the camera is not being read fast enough. At some point, the camera's internal buffer will be exceeded and frames will be lost. Frame loss is not the problem (it is desirable). The problem is this buffer can typically contain at least a second's worth of images. If you do not poll the camera fast enough, you will always receive a second old image. This is a huge problem when experiencing a large latency in the connection (which is a requirement of this competition). For this reason, the software polls the camera continuously in one thread and it only sends images when they are requested by the client. In this way, the client always gets the most recent image from the camera.

The Camera is also capable of holding multiple detector objects. These detectors are used to provide on-board image processing to potentially help the operator find rocks or dangers in the field. These detectors run with the following simple thread.

- Copy most recent image from camera
- Process image
- Send signal if something was found
- Repeat

The signal is an implementation of the publish/subscribe pattern typically used for asynchronous communication in systems.

The second implementation is a streaming camera used by the web server. The streaming camera does not have detectors. The only operation it has implemented is to get an image. The streaming camera connects to the physical camera interface and when an image is requested, it performs the following operations:

```
 - Check internal buffer time stamp
 - If internal buffer time stamp is X milliseconds old...
   - Then request a new image and change the internal buffer/time stamp
 - Return the internal buffer
```

In this way, multiple clients can request information from the streaming camera and it will not affect the internet connection of anything other than the server (who must process all of the requests).

While the technical aspects of the robot are obviously very important, there is another, perhaps unappreciated purpose for RHEA-1: to educate people about the field of robotics. If you were to ask someone on the street what they thought of when they thought of a robot, they would most likely only be able to respond with robots seen in popular culture, but the robotics being developed today are far more important and relevant than any fictional depiction in popular culture. Research being done today is paving the way for the entrance of consumer robotics into mainstream culture, which we believe is a major milestone to be achieved by the human race in the very near future. Already one can see simple robotics beginning to creep into our everyday lives, but this is only the beginning. Within the next few decades, experts predict that there will be a robot in every household.

As exciting as this is, people are still unaware of the advancement of research in the field of robotics. Research in this field is done primarily by highly trained professionals that have spent the majority of their adult lives researching robotics in laboratories and behind closed doors. As a result, the gap between an average person's knowledge of robotics and the actual state of the industry have grown so far apart that it is a very difficult task to keep people educated about robotics. Even if a professional were to explain to an average person how a robot works, the person would have no comprehension because modern robots are built on concepts and practices that require years of training to master.

These concepts are important for everyone to know because, at the rate at which the scientific community is advancing, more advanced concepts are finding their way more and more into mainstream culture. The robotics field today is behaving in much the same way that the early computer industry behaved. Originally, computers were so complex and required such advanced knowledge that the only people that were capable of operating these great machines were those that had spent years learning the concepts and techniques that computers were based upon. Today, the average person is very familiar with computers and many of the concepts computers are based upon. In the same way, years from now people will need to be able to interface with robotics the same way that we interface with computers today.

These are the circumstances under which RHEA-1 was built. We have worked diligently and tirelessly to build and produce a robot that not only fulfills all of the technical requirements necessary, but also incorporates a level of user interactivity that allows a person of any background to operate it easily. RHEA-1 has been designed to be compatible with numerous controlling mechanisms to support greater user functionality, and incorporates a live streaming webcam so that people at home can watch the robot maneuver through obstacles and complete tasks. In addition, RHEA-1 was predominantly designed using parts and components that are available to anyone at a very reasonable price. This was done so that if anyone watching RHEA-1 were to be inspired to build their own robot, they would be able to without encountering too much frustration and hardship.

All of these design features and many more were implemented on RHEA-1 in order to better facilitate our team's efforts to further educate people about the field of robotics. While a professional may not be able to coherently explain concepts to a normal person, it would be much easier for us to simply show these same concepts in action in RHEA-1. While RHEA-1 was built using concepts and techniques that require a very high level of expertise, many of its systems were designed in order to be easily accessible to people without any prior knowledge of robotics. In this way, RHEA-1 acts as a sort of bridge between the robotics industry and the rest of society.

We have used this idea and the design of RHEA-1 already in order to educate people about robotics. We believe that robotics is a very exciting field for everyone to learn about and this enthusiasm has shown in the presentations that we have made in order to further spread knowledge of robotics. From here in Washington D.C. to Scotland, we have been teaching everyone that we come across all about RHEA-1 and advancements in the field of robotics. Whether they be elementary school students or university professors, everyone has something to learn.

While our educational efforts have been somewhat limited due to the fact that RHEA-1 is still under development, it will be an extremely powerful tool in teaching people about robotics once it is complete. People will be able to look at our robot and will be able to understand how it works and on what concepts it is founded. This will inspire a new generation of learning students who are excited about robotics and the advancements being made in this field.