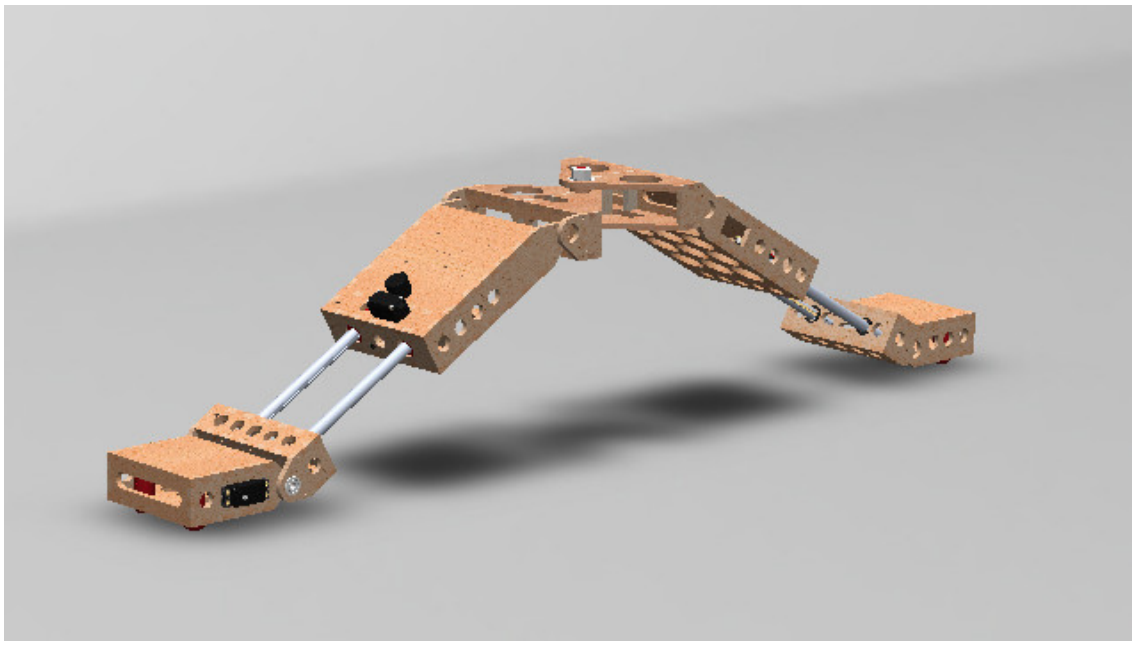


Space Jockey

A Mobile Robot Platform for Spacecraft Exterior Inspection & Maintenance

Final Report - Team B

December 12th, 2013

Brian Boyle, Nathaniel Chapman, Ardya Dipta Nandaviri, Songjie Zhong

THE ROBOTICS INSTITUTE

Abstract

The process of space launch and travel is a taxing process for both man and machine. Wear and tear incurred by spacecraft during launch and orbital operations can greatly increase costs by decreasing reliability and reusability, and can even lead to loss of life if not detected and addressed, as evidenced by the Space Shuttle Columbia Disaster in 2003. Thus, there is a definite need for spacecraft inspection and repair operations in orbit. The best current solution to this problem is met by astronauts performing Extra-Vehicular Activities (EVAs) or "Spacewalks" to perform manual inspections. This solution is versatile, intelligent, and mobile, but at immense costs and danger to astronauts. We propose building a small, portable robotic platform capable of traversing the exterior of spacecraft in orbit, and scanning for visual or thermal abnormalities on the hull. This solution could easily be safer, less expensive, and less time-consuming than astronaut EVA.

Table of Contents

Abstract 1
Table of Contents 2
1. Project Description 3
  1.1. User Needs 3
  1.2 Proposed System 3
2. System graphical representation/Use case 3
  2.1 System graphical representation 3
  2.2 Use Case 4
3. System-Level Requirements 5
  3.1. Mandatory Requirements 5
    3.1.1. Functional 5
    3.1.2. Non-Functional 5
  3.2. Desirable Requirements 6
    3.2.1. Functional 6
    3.2.2. Non-Functional 6
4. Functional Architecture 6
5. Physical Architecture 7
6. Current System Status 8
  6.1. Fall-semester targeted system requirements 8
  6.2. Subsystem breakdown 9
    6.2.1. Space Jockey Robot Subsystem 10
    6.2.2 Command Center Subsystem 14
  6.3 Modeling, analysis, and testing 16
    6.3.1 Adhesive foot attachment testing 16
    6.3.2 Gait Model 16
    6.3.3 Gait State Model 18
  6.4. Performance Evaluation Against the Fall Validation Experiment 19
  6.5. Strengths and Weaknesses 19
7. Conclusions 20
  7.1. Lessons Learned 20
  7.2. Test Plan 21
    7.2.1 Capability Milestones for these Spring-semester Progress Reviews (PR) 21
    7.2.2 Test Conditions: 21
    7.2.3 Test Procedures 21
    7.2.4 Quantitative performance metrics: 22
  7.3. Schedule and Budget Status 22
    7.3.1 Spring Semester Schedule 22
    7.3.2 Current Budget Status 23
  7.4. Risk Management 25

1. Project Description

The process of space launch and travel is a taxing process for both humans and machines. Wear and tear incurred by spacecraft during launch and orbital operations can greatly increase costs by decreasing reliability and reusability. In extreme cases, even typical wear and tear can lead to loss of life if undetected, as evidenced by the Space Shuttle Columbia Disaster in 2003 (figure 1).

Figure 1. Columbia Space Shuttle disaster

Thus, there is a definite need for spacecraft inspection and repair operations in orbit. Currently, this need is met by astronauts performing Extra-Vehicular Activity operations (EVAs) to perform manual inspections. However, this solution adds risk to the mission, consumes valuable in-flight resources like power and fuel, and is significantly time-consuming and expensive.

1.1. User Needs

The Space Jockey mobile robot is intended for use by organizations owning and operating long-term or reusable spacecraft that require regular inspection and maintenance while in orbit. In particular, the robot shall be used as a safer and more cost-effective solution to hull maintenance than astronaut EVAs. Thus, the intended users of our robot would be astronauts or Earth-based technicians who might otherwise commission an EVA mission to accomplish spacecraft maintenance tasks.

To decrease or assist the need for astronaut EVA, the robot must be able to:

1. Secure itself to a spacecraft hull using mechanisms that function without the aid of gravity or atmospheric pressure.
2. Accept commands from astronauts aboard the craft or other operators on Earth.
3. Traverse in such a manner that a large proportion of the hull's surface area is reachable.
4. Visually inspect the craft surface and report its findings to a human operator
5. Function reliably in a space environment for a useful period of time.
6. Maintain environmental awareness, and avoid collisions with obstacles that could damage either the robot or the craft itself.
7. Operate tools in order to perform repairs or modifications to the craft.

1.2 Proposed System

We propose building a small, portable robotic platform capable of traversing the exterior of a spacecraft while in orbit, and scanning for visual or thermal abnormalities on the hull. Such a system would be teleoperated remotely to assist or supplant astronauts in EVA, such as by performing visual inspections, operating tools, or acting in a support role to an astronaut in EVA. This could provide a more cost effective and safer option for mission planners.

2. System graphical representation/Use case

2.1 System graphical representation

The graphical depiction in figure 2 demonstrates the basic mission phases of a robotic inspection/maintenance task. An operator inside the craft or on Earth commands the robot via a

computer console. The robot first deploys (1) onto the surface of the craft, then traverses (2) the surface of the craft. While traversing, the robot may stop to perform inspection or repair tasks (3), and concludes its mission by returning (4) to its point of origin and stowing (5) until its next mission.

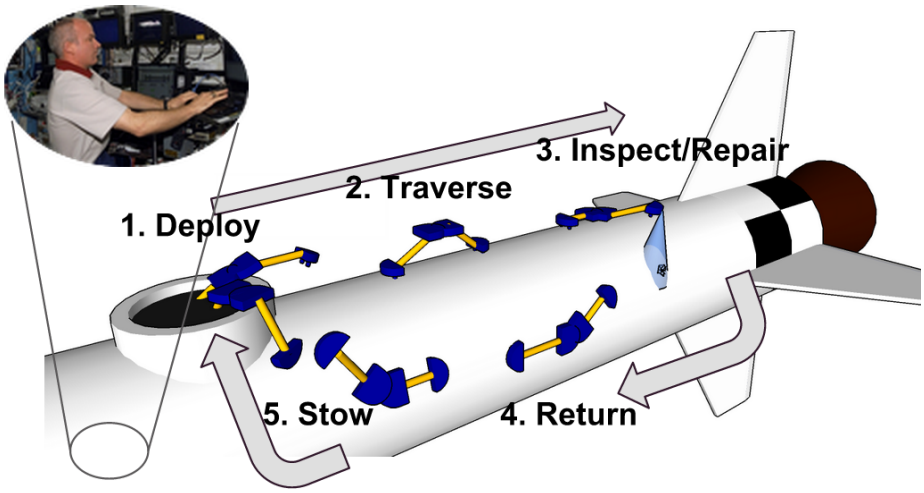


Figure 2. Use case of Space Jockey System

2.2 Use Case

On May 30th, 2018 NASA Mission control registered an atmosphere level warning from the Quest Airlock Module onboard the International Space Station. Initial assessment of the atmosphere losses suggested a fault with the airlock control and recycling mechanisms, but additional diagnostics failed to identify an equipment fault. NASA archives recalled a communication satellite collision taking place on a similar inclination in 1978, and it was hypothesized that the ISS may be orbiting near the resulting debris field. This led mission controllers to believe that the module may have suffered an impact event with an untraceable 2-3cm piece of debris. The module was immediately sealed to ensure crew safety.

Mission control began discussing repair options for the module, agreeing that proximity to the debris field posed an elevated risk to an EVA operation. Additionally, the Canadarm2, often used for transporting tools and astronauts during EVAs, was incapable of being retrofit with the necessary repair equipment from inside the station. It was decided to deploy the newly acquired Space Jockey robot from the smaller experiment airlock onboard the Japanese Experiment Module “Kibo”, near the desired inspection site.

Mission Cmdr. Jebediah Kerman and his twin brother Bob were tasked to unpack the robot from its storage module and fitted it with a temporary hull sealing device. The roughly ½ meter cubed unit was then prepped, powered up and tested, before being deployed through the undersized experiment airlock aboard Kibo. A path was planned for Space Jockey by the station’s onboard computer and wirelessly transmitted to the robot’s computer, enabling it to navigate to the Quest Airlock Module.

After two hours of teleoperated inspection tasks, the operating crew found that a small impactor had indeed punctured the prograde side of the airlock module, adjacent to the crew lock.

Visual inspection revealed that a 3-inch hole had been torn through the hull's outer shielding plates, and thermal imaging revealed that a hairline fracture had punctured the inner pressure vessel, leading to the loss of atmosphere from that module. The robot deployed sealing foam to the area as a temporary measure, restoring the module to provisional operating status. A return path was generated and transmitted to the robot, and it returned to Kibo's experimental airlock for retrieval and storage by the crew members.

After the robot's intervention, the crew was able to apply hull sealant to the interior of the pressure vessel, permanently sealing the leak. To complete repairs, replacement shielding plates for the module were included in the next station service mission, and an EVA scheduled for external repairs to mitigate future impact dangers in the area. The use of the space jockey robot provided a critical first step in the repair process, and allowed the station to be fully repaired with a minimum level of danger to the crew.

3. System-Level Requirements

Requirements for the Space Jockey system are listed in the following sections, divided into mandatory and desirable requirements. Both critical and desirable requirements are noted, and Technical Performance Metrics (TPM) are noted where applicable.

3.1. Mandatory Requirements

3.1.1. Functional

Table 1. Mandatory Functional Requirements

#	Code	Description
1	ATTACH	Maintain attachment to the hull* TPM: Supports own weight on a 1G vertical surface
2	TRAV	Traverse the craft at a reasonable speed TPM: 10 meters per hour
3	PLANE	Plane Transitions TPM: 60-degree transitions
4	SENSE	Visual feedback from cameras, joint encoders, and foot sensors
5	*FOOT	Detect footholds with actionable accuracy

3.1.2. Non-Functional

Table 2. Mandatory Non-Functional Requirements

#	Code	Description
1	BATT	Battery Powered
2	EARTH	System can be validated in terrestrial conditions
3	MASS	As light-weight as possible to reduce launch requirements TPM: <10kg
4	SIZE	Minimal volume for efficient storage TPM: 8 Cubesats (10cm cube)

5	COST	Total system cost must not Exceed $4000

3.2. Desirable Requirements
3.2.1. Functional

Table 3. Desirable Functional Requirements

#	Code	Description
1	MANIP	Use a rotary tool to tighten a fastener or perform other maintenance function
2	TEMP	Thermal Imaging capability
3	SPACE	Develop plan to convert into a space-ready system
4	DETECT	Autonomous identification of surface flaws
5	PLANE2	Plane Transitions (90-degree)

3.2.2. Non-Functional

Table 4. Desirable Non-Functional Requirements

#	Code	Description
1	WIFI	Completely wireless operation
2	INFRA	Minimal Spacecraft Support Infrastructure
3	ATTACH2	Maintain attachment to the hull* TPM: Supports own weight on a 1G **inverted** surface

4. Functional Architecture

The Space Jockey system consists of two major subsystems, the mobile robot and the command center. The robot traverses the exterior of the hull and performs inspection and manipulation tasks, while the command center allows for a human operator to exert control over the robot's efforts and monitor its state.

The robot is primarily teleoperated, with some autonomy and path planning features, a diagram of the full functional architecture is depicted in figure 3. The robot captures images using its cameras transmits them to the command center over a radio link. The command center receives the signal from the robot and processes the images, comparing them against previously known inspection data to detect any flaws. If significant deviations are discovered, the user is notified so that a repair plan may be generated. The user may also monitor the image feedback directly.

The operator may issue directives through the command center, which then generates a movement plan based on its internal hull map and the robot's current position. After a path is generated, these instructions are transmitted to the robot via the radio link, and the robot

performs the desired maneuvers. The robot's progress may be monitored through the command center, and its actions halted or modified as deemed necessary by the operator.

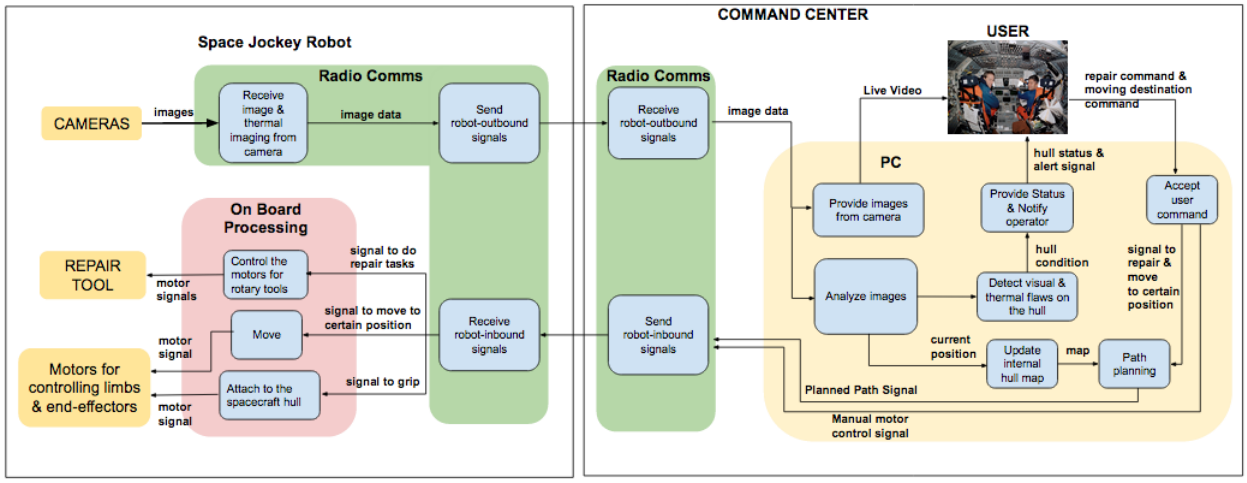


Figure 3. Functional architecture

5. Physical Architecture

The Space Jockey consists of two major subsystems: the command center and the robot. The two components connect to one another using 2.4GHz XBee radio serial modules. The robot uses an Arduino Due as the primary controller, and implements motor control using an I2C PWM servo driver board. A Linux powered PC running ROS is used as the command center to process and log camera data, plan paths and queue robot movements. User commands are input using the keyboard/mouse and a custom Qt frontend attached to ROS. The full physical architecture is shown in figure 4.

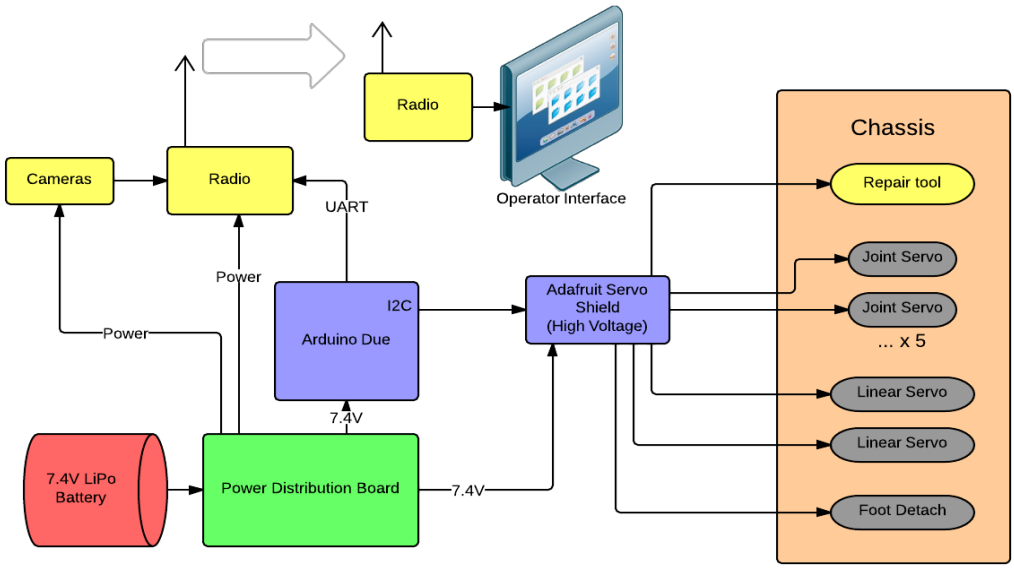


Figure 4. Physical Architecture Diagram (note that the components designated in yellow are those planned for the Spring semester)

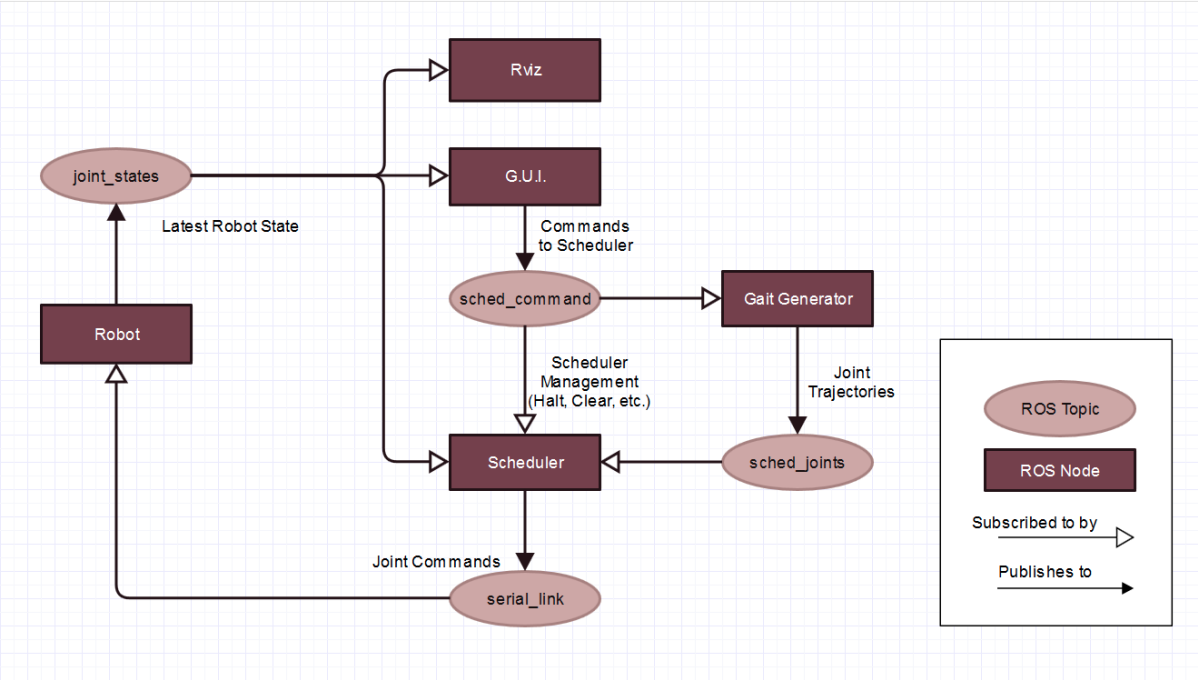


Figure 5. ROS Software architecture

The information flow within the control system is depicted in figure 5. All information between processes is exchanged using ROS communication. Rviz provides a visualization of the robot's kinematic state, while the GUI allows the user to issue commands to the gait generator and scheduler. The gait generator provides joint trajectories between robot states to the scheduler, which stores these in a queue and transmits joint states via serial USB link to the robot, which executes them, and finally informs Rviz and GUI of the most recently executed joint command.

6. Current System Status

6.1. Fall-semester targeted system requirements

Table 5. Fall-semester targeted system requirements

Code	Description
ATTACH	Maintain attachment to the hull TPM: Supports own weight on a 1G vertical surface
TRAV	Traverse the craft at a reasonable speed TPM: 10 meters per hour
BATT	Battery Powered
EARTH	System can be validated in terrestrial conditions
MASS	As light-weight as possible to reduce launch requirements TPM: <10kg

This semester, we met all of our targeted system requirements except for our hull attachment mechanism. In testing, we found that our current adhesive foot design was incapable of supporting the mass of the robot. On account of this, we demonstrated the robot on a horizontal surface for our fall validation experiment. For traverse speed targets, the robot was able to move at roughly 1 meter per minute at max speed, far exceeding the 10m/h TPM. Additionally, the robot's current weight is 3.1kg, far less than the targeted 10kg, with additional weight reductions planned for future development.

6.2. Subsystem breakdown

The system's subsystem breakdown is shown in figure 6. Note that command center subsystems constitute software processes implemented in ROS, while those of the robot are primarily hardware components. Peripherals (cameras, IMUs), manipulation, and autonomy (localization, planning) subsystems are planned for implementation in the Spring semester.

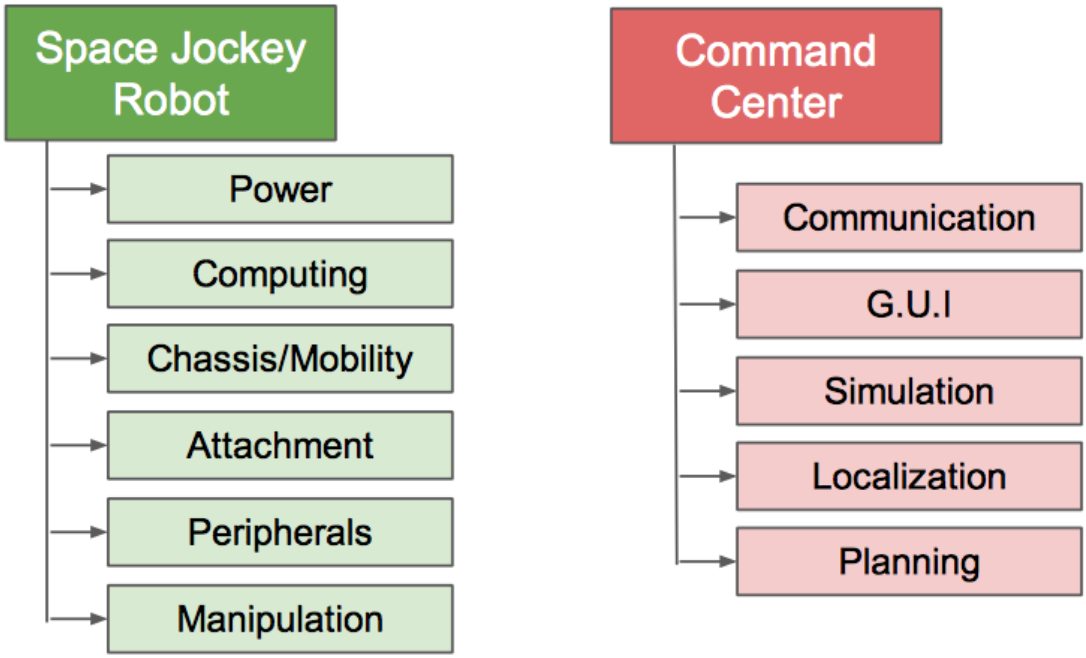


Figure 6. Subsystem breakdown

6.2.1. Space Jockey Robot Subsystem

Power Subsystem

The robot is powered by a 7.4V hobby Lipo battery. All power is routed through the power distribution board, which provides two unregulated 7.4V servo channels rated at 10 A and one 7.4V channel for the Arduino power at 1A. The completed board is shown in figure 7 and its schematic in figure 8.

Since the full requirements of the power system will not be known until Spring semester, the power distribution board for the robot was designed to be future proof. This is accomplished with three additional output channels which can be populated with standard linear regulators to provide a variety of output voltages. These will be used in the future to supply the sensors or cameras needed to accomplish the Spring semester goals.

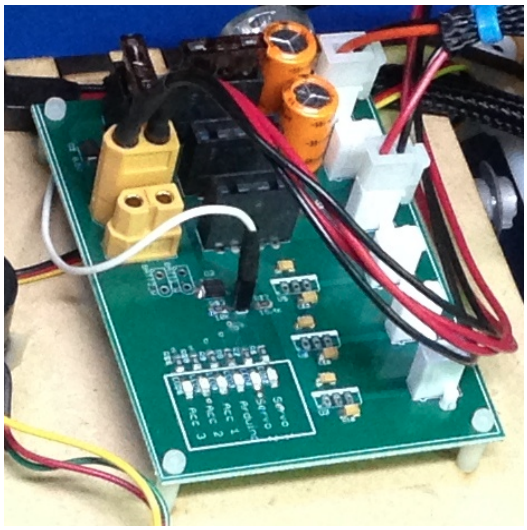


Figure 7. Power distribution board

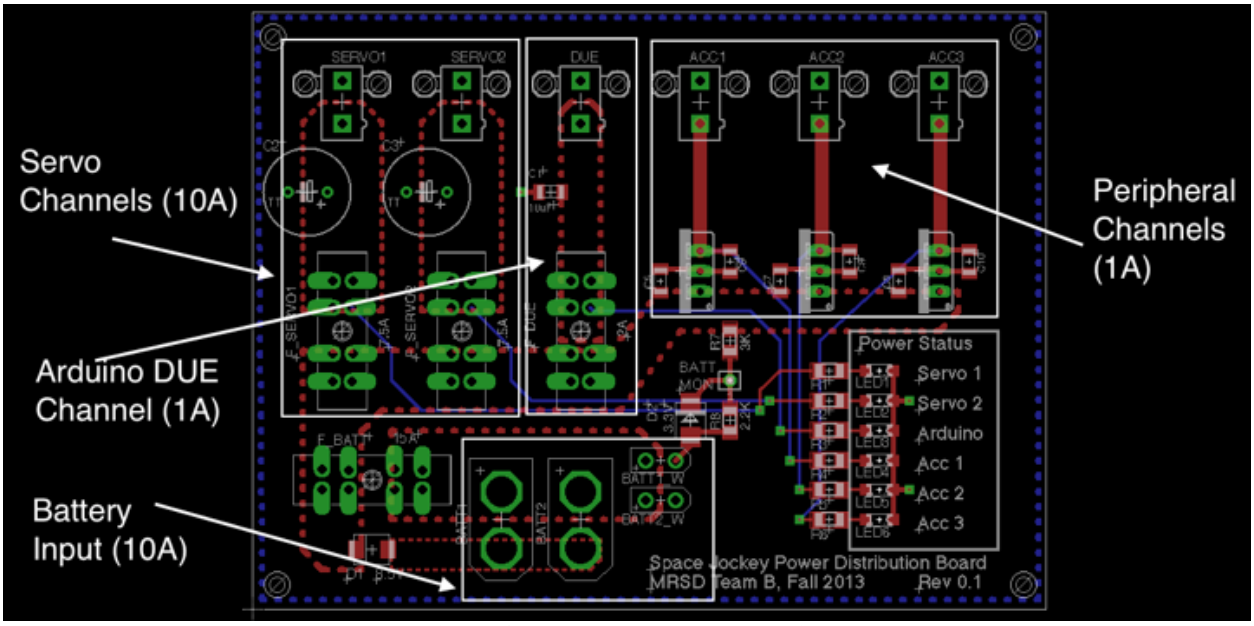


Figure 8. Power board layout design

Computing Subsystem

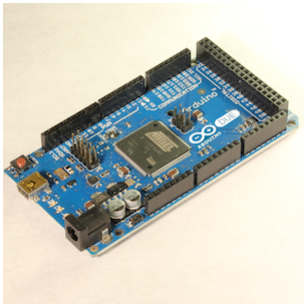


Figure 9. Arduino DUE

The microcontroller for our chassis is an Arduino Due (figure 9). The system is controlled by the Arduino when attached to a laptop running the operator interface. The Due uses a 32 bit ARM core chip, allowing for quick floating point calculations and larger math operations onboard. The Due is small and lightweight, and provides ample computing power to implement a simple ROS node and serial communications, making it an ideal choice for this project.

The interface between the microcontrollerer and servo motors is the Adafruit Industries 16 Channel Servo/PWM controller shield (figure 10). The shield's 12-bit interface allows a theoretical servo control accuracy better than ±0.1°, although the servos themselves are only accurate to ±0.3°. It communicates over an I2C interface, making it easy to offload PWM servo control tasks for our robot, allowing finer granularity than could be achieved using the Arduino alone.

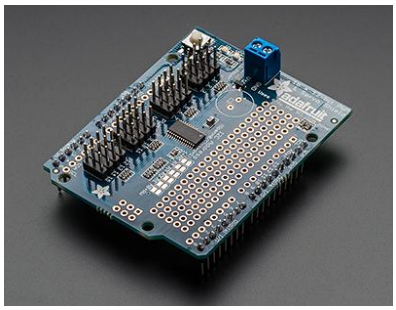


Figure 10. Adafruit servo controller shield

For radio communications, we plan to use XBee Pro 60mW radio modules, which implement the IEEE 802.15.4 standard, to act in the Spring semester as a drop-in replacement for the current USB connection.

Chassis Design

Figure 11 shows the design of the robot's chassis in Solidworks. Because we were not able to find a linear actuator that met our weight goals and requirements, we designed and built custom linear actuators using modified hobby servos and a rack-and-pinion gear. This approach worked well, and helped to keep the total mass of the robot under 5 Kg.

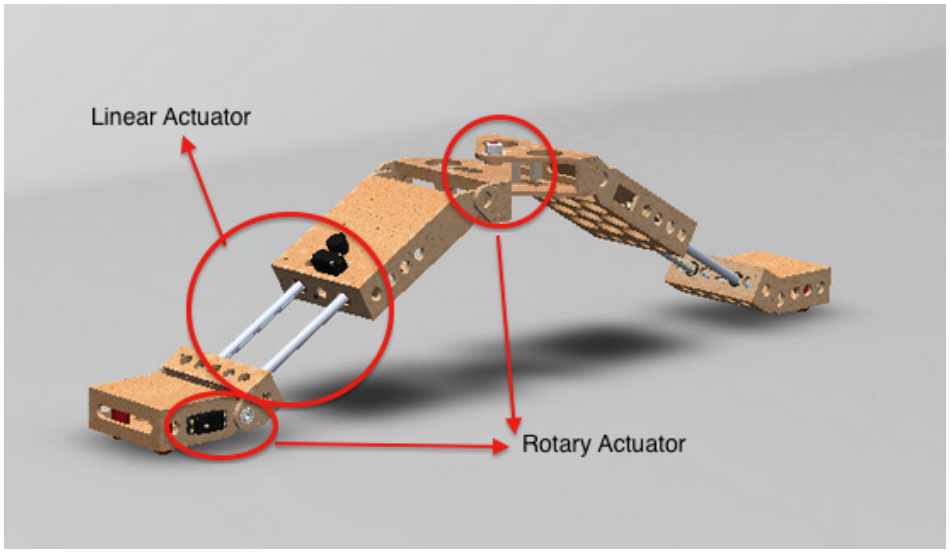


Figure 11. Solidworks of the robot's chassis

The chassis was fabricated using 1/4" MDF, machined using a laser cutter. This fabrication method allows rapid redesign and prototype iteration turnaround. Using a laser cutter, the fabrication of an entire chassis parts set takes only about 1.5 hours, and the MDF material is only around $5 for a sheet, so the cost for iterative development is very low, which has proven to be a great strength of our development. An example laser cut parts plan is shown in figure 12.

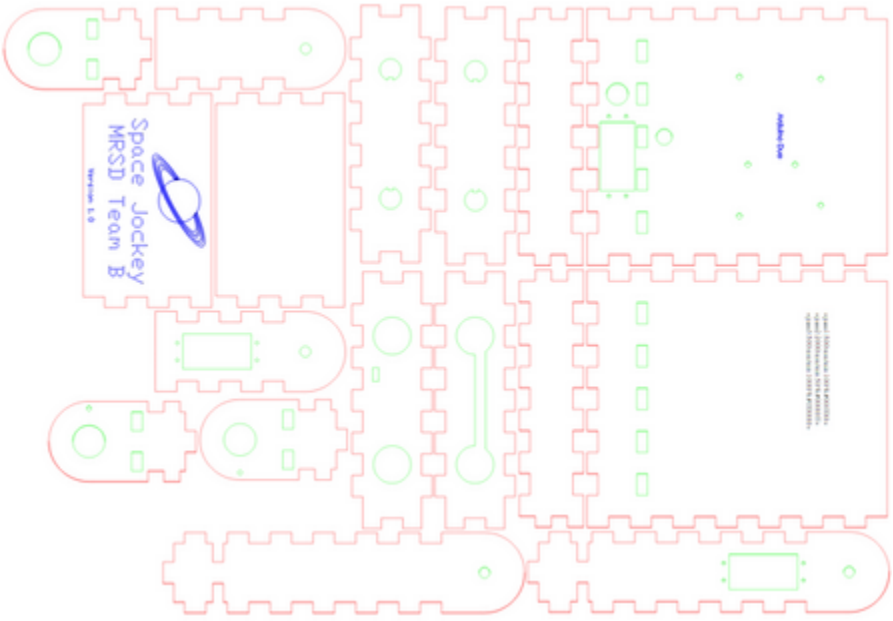


Figure 12. Example of laser cutting plan, ready for fabrication. Red lines are external edges, green lines are internal edges, and blue lines are engraved annotations and graphics

The chassis is actuated using Hitec hobby servos. For the shoulder and wrist joints, where high torque was needed, Hitec HS-5585MH digital metal-gear high-voltage servos were used (See figure 13). In addition to the higher torques and better control accuracy of the digital servos, these high-voltage models may be powered directly from the battery, simplifying power distribution and increasing power efficiency and battery life.

Figure 13. Hitec HS-5585MH

For the linear actuators and center segment joint, where less torque was required, HS-5496MH servos are used. These have the same advantages as the HS-5585MH, but are much more affordable. For the linear actuators, these were modified and attached to an external feedback potentiometer, allowing them to be calibrated and adjusted to provide maximum possible accuracy over their specific linear range.

All servos were calibrated using the Hitec HPP-21 digital servo controller programmer (figure 14) which allows precise adjustment of the range and accuracy of the servos, and calibration of the robot's position in hardware, obviating the need for complex software calibration and configuration joint angles. The programmer also allows the setting of fallback positions and stall voltage dropouts, to protect the power system and chassis in case of collisions with a physical obstacles or loss of

Figure 14. Hitec HPP-21 digital servo control programmer

connection to the command center.

Attachment Mechanism

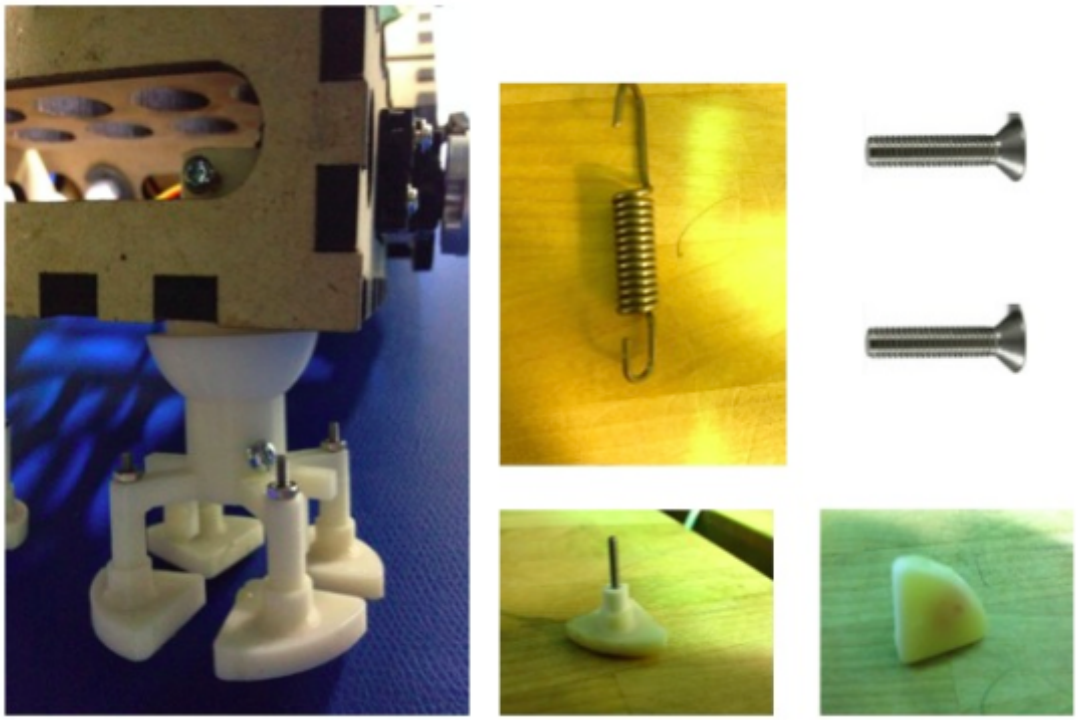


Figure 15. Detail of adhesive foot mechanis,. showing conformant ball joint spring, attachment hardware and removable segments

Attached to the bottom of each robot segment are one or more legs which serve to maintain attachment to the spacecraft hull, pictured in figure 15. Each leg is composed of a conformant ball-joint which allows attachment to irregular surfaces, a foot mechanism, and removable adhesive toe pads which attach to the foot using 4-40 machine screws. The use of detachable footpads significantly improves the turnaround rate for iterating pad shapes and experimenting with different techniques for casting the adhesive surface.

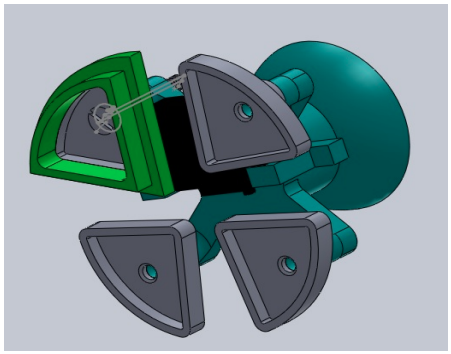


Figure 16. Solidworks model of the foot design

The “ankle” of each leg consists of a compliant ball and socket joint which are held together by a 4-40 screw which maintains tension between the top of the foot and the base of the “shin”. The physical constraints of the leg mounting limits flexion in the sagittal and coronal planes to roughly 15 degrees, and the tension of the spring allows flexion in the axial plane of at least 90 degrees.

For the adhesive foot pad, we used Smooth-On Vytaflex 10, a very soft (Shore 10A hardness) urethane based casting compound recommended by professor Metin Sitti for this purpose. Casting the adhesive pads presented engineering challenges in itself. Initial attempts entailed 3D printing a mold for the entire foot, which was then partially filled with V-10 material onto which foot pads were placed. After the material hardened, removal of the foot from the mold proved to be extremely difficult. In the worst case, the entire 3D printed foot pad was torn from the foot, an issue which led us to produce detachable toe pads.

Under these conditions, the adhesive pad often stuck so firmly to the walls of the mold that it was impossible to remove, and attempts at removal often tore the adhesive from the pad itself. Learning from this, we added retaining walls to the edges of the footpad, so that the strength of the bond between the adhesive and the foot would also benefit from shear force resistance.

Even when the pad was successfully removed, the adhesive surface was often torn, scarred, or otherwise damaged in ways that compromised its grip on surfaces. This led to our final revelation in casting, which was to remove the bottom of the mold and flip the entire rig to face upward, resulting in the smoothest-yet pad surface. Casts using this technique demonstrated some curvature of the adhesive pad due to surface tension, but this caused no apparent issues in bond quality once the foot was firmly pressed against a surface.

6.2.2 Command Center Subsystem

The command center software is comprised of three primary processes, run on a single computer connected to the robot by serial USB tether. All processes are implemented in Python and communicate through ROS messaging unless otherwise noted. The following is an expanded description of the software architecture depicted in figure 5.

The GUI consists of a widget panel developed with QtDesigner and implemented as a custom plugin for RQT, ROS's Qt-based framework for GUI development. This custom-built GUI works in tandem with Rviz to provide a control interface with the robot and its visual representation, respectively. As the user interacts with the GUI, it issues commands to a gait generator and scheduler.

The gait generator receives commands from the GUI in form of state transitions (e.g. lift the retracted front segment) and generates joint trajectories connecting the current state to the new one states. This is accomplished by interpolating segment positions between these states and inferring the necessary joint positions necessary to meet them through inverse kinematics.

Two parameterized variables of the gait determine its shape, the desired clearance between lifted segments and the ground, and the maximum extension of linear actuators during an entire gait cycle. These are stored within the gait generator instance but can be changed via commands from the GUI. The resolution of interpolation by the gait generator is determined by a set of variables corresponding to lift/place, extend/retract, and turn/retract motions. These variables can be modified from the GUI , giving the user fine control over the speed and smoothness of particular movement types.

The scheduler is responsible for organizing joint trajectories and delivering them to the robot through the serial USB link. To this end, incoming joint states are pushed onto a queue, while a constant loop runs to pop one at a time and issue them to the robot. The delay between each joint state pop is governed by a local variable which can be modified when commanded to by the GUI. The GUI is also capable of commanding the scheduler to halt, resume, and clear its queue as necessary, and of issuing custom joint configurations directly to the scheduler.

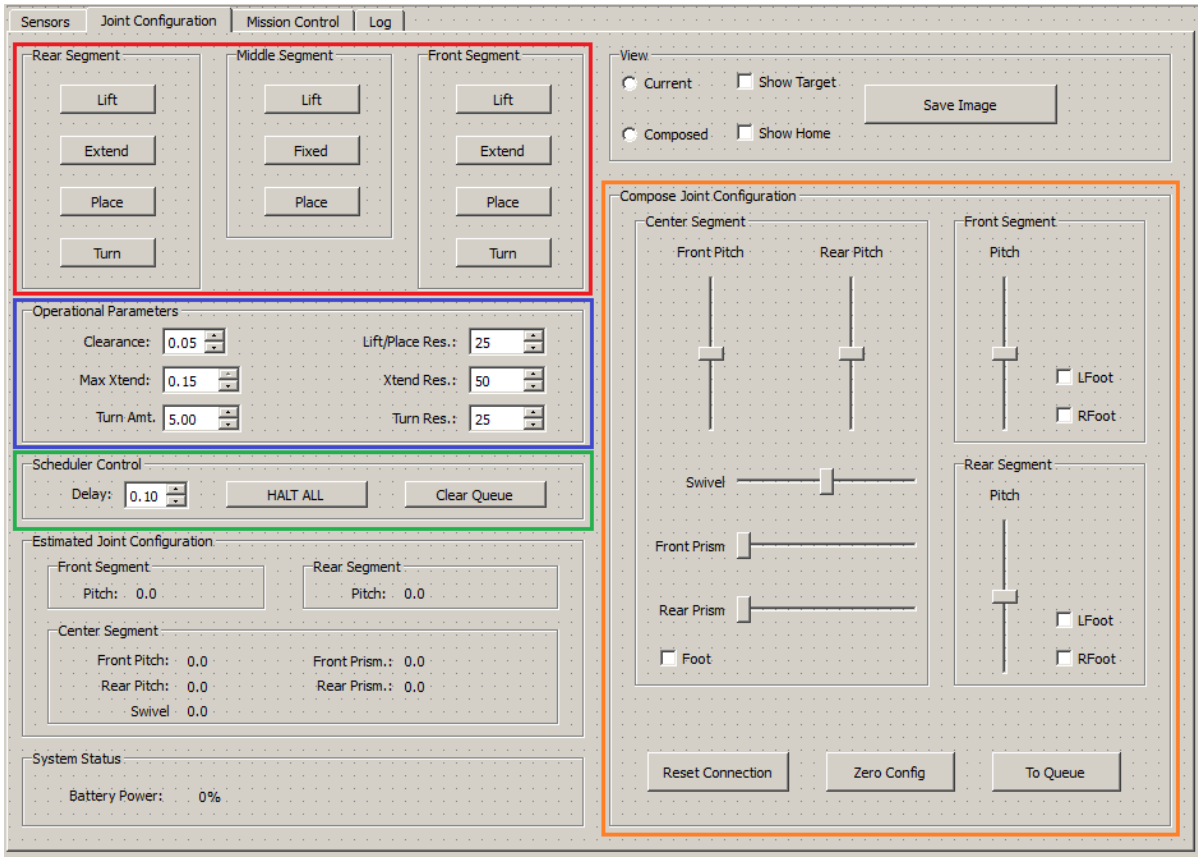


Figure 17. The graphical user interface, with colored boxes delineating control groupings

The GUI is depicted above in figure 17. The buttons located in the red box are used to guide the robot through the various states in gait and turn cycles, and are enabled and disabled according to the state the robot is in. The widgets in the blue box are used for altering the parameters used by the gait generator to determine and interpolate robot configurations. Scheduler controls are featured in the green box, and the widgets in the orange box are used to manipulate the Rviz robot visualization, and issuing custom joint states directly to the scheduler for testing and debugging.

6.3 Modeling, analysis, and testing
6.3.1 Adhesive foot attachment testing

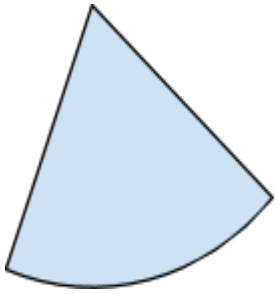


Figure 18. Toe pad area

Figure 19. Adhesive foot load testing

Bench tests (See figure 19**)** of the V10 material cast on each foot demonstrated a vertical holding capacity of 0.6 lbs. The surface area of each foot (figure 18) is $(\pi R^2*(80/360))*4 = 11cm^2$.

Given the current number and distribution of legs, this is insufficient to support the entire robot on the vertical surface. Further development and refinement of the attachment mechanism is planned for the Spring, and in the event of a complete failure to leverage adhesive strategies we plan to implement a hardpoint attachment mechanism.

6.3.2 Gait Model

The gait sequence that propels the robot forward and backward consists of a simple sequence whereby each segment is moved one after another. For each segment, a lifting motion is performed, followed by an extension (or retraction, in the case of the trailing segment), and a placing motion. Moving segments in straight horizontal and vertical lines results in a square-shaped motion path followed by each segment, as depicted for the front segment in figure 20.

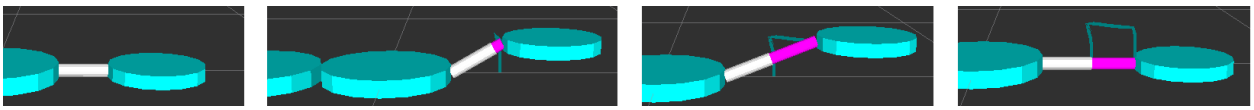


Figure 20. The front segment performing a square step forward

The determination of joint states during the square gait is mathematically straightforward. Two key parameters determine the dimensions of the square path traced by each segment, the maximum extension of linear actuators and the desired clearance between a lifted segment and the ground. These parameters imply the goal states of each state transition, and at each goal state the joint angles and actuator extension can be determined by simple trigonometry on a right triangle.

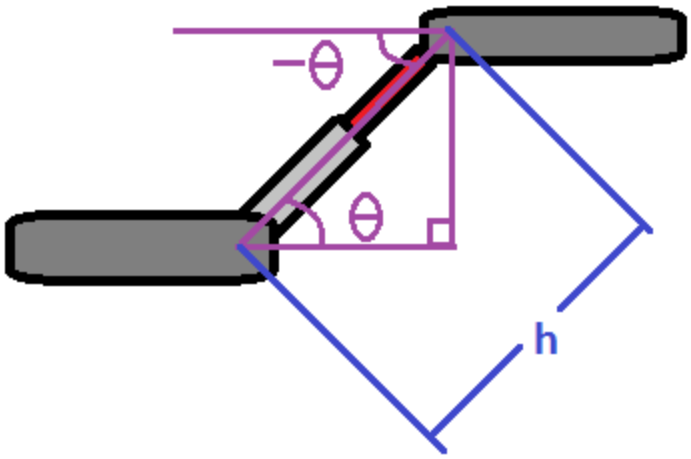


Figure 21. Relevant parameters for segment motion

For example, for a lifted front segment in a retracted position (See figure 21), the horizontal distance from the center segment to the front segment is given by the retracted length of the linear actuator, and the height is given by the clearance parameter. These two lengths are sufficient to find the angles of the two revolute joints and the final extension of the linear actuator once raised. During lifted extension, a hypotenuse is formed by the fully extended actuator, and a second side is formed by the clearance value. The length of the third side can then be used during the place maneuver to act as the resting distance between the two segments. Movement of the middle segment is equivalent to extending one end while retracting another.

6.3.3 Gait State Model

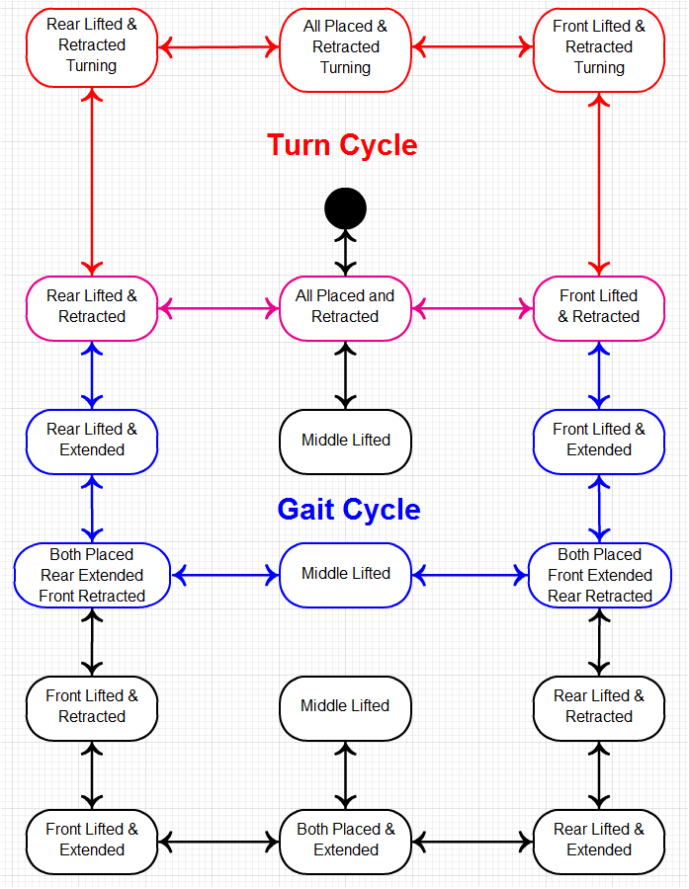


Figure 22. Square gait state diagram

The state of the robot during the square gait is governed by three simple rules:

1. Only one segment may be lifted at any time
2. End segments may be characterized as extended or retracted at the end of each transition
3. Linear actuators may not extend or be extended during turn maneuvers

These rules imply a set of states that are permutations of each of three segments being either lifted or placed, extended or retracted, and either turned or straightened. When diagrammed as in figure 22, the interactions between states are straightforward and each state has at most three available transitions. The Gait Cycle, marked in blue, forms a loop that provides forward motion in one direction and reverse in the other, and likewise for the Turn Cycle marked in red. States and transitions marked in purple constitute the start and end sequences for both the gait and turning maneuvers.

6.4. Performance Evaluation Against the Fall Validation Experiment

The results of the Fall Validation Experiment are summarized in table 6 below. The test procedure is listed in sequence with a brief description of each step and comment on the result. More in-depth discussion on these results may be found in the following section.

Table 6. Summary table of validation experiment results

Test	Description	Result
1	Robot powers on & GUI is started	Robot powered on with no issues
2	GUI will display sensor data from the robot	Displayed battery status
3	Robot commanded to pre-configured stances that demonstrate actuation of all motors	Demonstrated during gait
4	Robot commanded to move forward and backward and demonstrate such motions [TPM: 10 meters per hour]	Walked forward and backward successfully at an average speed of 1 meter in 10 seconds
5	Robot will maintain or recover its stance in response to external disturbances	Robot maintained configuration but was subject to tipping over
6	Robot commanded to execute a 45 degree turn while moving forward, and will demonstrate such motion [TPM: Turn within 5 degrees]	Robot was able to execute turns but degree accuracy failed to meet TPM

6.5. Strengths and Weaknesses

Testing and integration trials made clear the strengths and weaknesses of the system as developed so far, summarized in table 7 and 8. A major weakness of the system was revealed by intermittent communication dropouts suffered during the fall validation experiment. It is unclear at this time whether this is caused by power brownouts or by communication issues with the command center. Another weakness is that the command center GUI is reliable and highly configurable, but lacks user friendliness and would likely be confusing for an uninitiated tester. The chassis is very lightweight, thanks to its MDF construction material, but there are multiple sources of flex that cause the robot to sag during operation, often dragging its feet on the ground.

The attachment mechanism is the most pressing weakness of the system. Though on the horizontal surface it was able to attach to the ground, the lifting of segments as a means of detachment was unreliable and error prone. In addition, the complaint ball joint in the ankle seems to impede detachment during segment lift actions. From these findings, it's clear that a distinct detachment mechanism will be necessary to achieve reliable lifting of each segment. Additionally, each leg currently unable to hold more than 0.6lbs (0.272 kg) of mass, which is insufficient to support our robot on a vertical surface, this will require drastic steps to rectify this issue in the Spring semester.

Table 7. Robot subsystem strengths and weaknesses

	Strengths	Weaknesses
Power	Able to Power all Motors	Anomalous Brownouts?
Computing	Simple ROS Communication	Serial & I2C Comms Problem
Chassis/ Mobility	Lightweight, Rapid Fabrication	Significant Flex
Attachment	Refined Molding Techniques	Weak Hold Attach/Detach Mechanisms

Table 8. Command center subsystem strengths and weaknesses

	Strengths	Weaknesses
Communication	Simple ROS Communication, Scheduler, Gait Generator	Occasional lost packets Occasional connection loss
G.U.I	Reliable, dummy-proof, highly configurable	"Engineer's GUI" Lacks Spring Capabilities
Simulation	Clean kinematic representation	No physical dynamics

7. Conclusions

7.1. Lessons Learned

- 3d printing proved to be a highly effective means for rapidly prototyping complex parts, though with strength limitations and exacting requirements of the printer calibration. We'll continue to leverage the experience we've gathered in the future, but focus on creating stronger components.
- Another valuable rapid prototyping method was laser-cutting. We've established an efficient process for fabricating using this method, and will continue in the Spring Semester.
- Late integration of software with hardware made it very difficult to adjust the software for the issues of flex discovered in the completed prototype. In the future, we plan to parallelize development more effectively to avoid this, implement a more sophisticated simulation, and may even maintain multiple prototypes.
- Another reason the completed prototype performed with unexpected flex was the failure to integrate physical dynamics into our simulation. This lesson could be best summarized as, "Rigid bodies are not indeed rigid." We plan to do modelling of physical dynamics to simulate flex in the Spring to avoid repeating this mistake.
- Experiences with shipping and supplies indicated a set of clearly dependable suppliers, namely McMaster-Carr, Servo City, Amazon, and Adafruit Industries. We'll prioritize these shippers in the Spring and continue ordering parts well in advance of planned integration.

7.2. Test Plan

7.2.1 Capability Milestones for these Spring-semester Progress Reviews (PR)

The capability milestones for the Spring Progress Reviews have been laid out to space development of key functionalities over the months of January through the beginning of March. The first two weeks of April are left as a buffer to account for unexpected delays, or under optimistic conditions provide time for the implementation of desirable extra features. A more detailed account of the planned goals to be met for each review may be found in table 9.

Table 9. Progress review dates and tasks to be completed

Progress Review	Task Description
Jan. 20	Simulation complete, Prototype redesign begun, Cameras, WIFI, IMU acquired, New adhesive foot prototype
Feb. 13	Mapping representation, Cameras, WIFI, IMU integrated, Final chassis designed, Hardpoint design begun if needed
Feb. 27	New hardware/software integration, Test environment designed, Hardpoints integrated
March 20	Demo localization and path planning in simulation, Final chassis fabricated, Test environment materials acquired
April 3	Demo waypoint functionality, Test environment assembled
April 17	*TBD: Catch up buffer or Desirable features*

7.2.2 Test Conditions:

Test rehearsals will be held in NSH Level B and the Spring demo shall be conducted in the NSH Atrium. Required equipment consists of the Robot, command center, spacecraft analogue structure with plane changes, scaffolding, and a safety tether attached to the robot. The spacecraft analogue will be suspended from the 4th floor of the NSH Atrium. The estimated operating area for the entire demo is 12'x12'.

7.2.3 Test Procedures

At the end of spring semester, we intend to have a completed hull-crawling prototype robot. It shall be tested by traversing about a suspended spacecraft surface analogue while sending video and sensor feedback to a human operator. The robot's destinations will be directed by operator, and the command console will generate paths for the robot, which it will then transmit to the robot for execution. The total distance travelled will be equal or greater to 10 times the robot's length with legs fully extended. Additional subsystem tests will be performed as necessary to demonstrate the meeting of all critical system-level requirements. If within the final scope of the project, the robot shall also demonstrate use of a rotary tool and/or thermal imaging.

The following is a detailed procedure for the Spring Validation Experiment:

1. Robot is powered on and GUI indicates communication and sensor feedback
2. Robot is commanded to move by teleoperation and begins inspection
3. Waypoints are set using GUI
4. Path planning generates a motion plan to reach said waypoints
5. Robot will autonomously travel between the waypoints while updating sensor map [TPM: 10m/hr]
6. At least one waypoint shall require a plane transition [TPM: 60 degrees]

Desirable extras if time permits their implementation:

1. During teleoperation, the robot shall navigate to and tighten a fastener using a rotary tool [TPM: Approached and tightened within 10 minutes]
2. In the course of mapping, the command console will identify and alert the user to a flaws in the hull [TPM: 80% Identification success less than four false positives per hour]

7.2.4 Quantitative performance metrics:

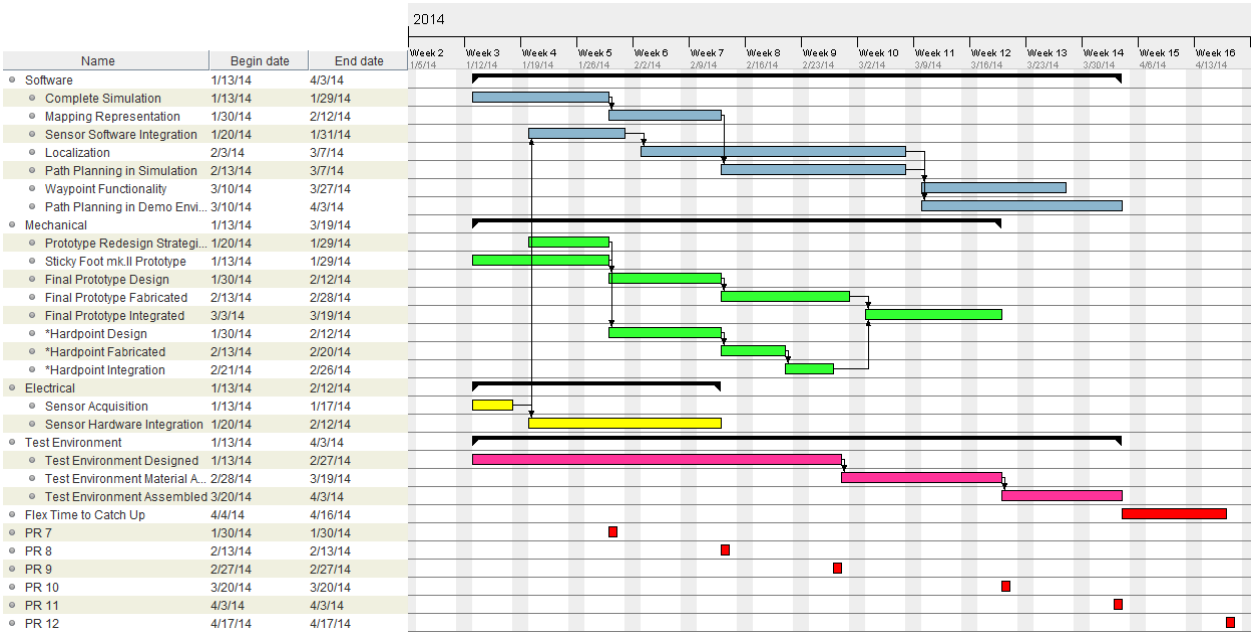
Table 10. Performance metrics for Fall Validation Experiment

Code	Performance Metric	Result
ATTACH	The robot shall be able to attach on a 1G vertical surface and support its own weight	Failed
TRAV	The robot shall traverse with speed 10m/hour	Successful (speed 60m/hour)
MASS	The robot shall have weight < 10kg	Successful (weight 3.1 kg)
BATT	The robot shall be battery powered	Successful
EARTH	The robot can be validated in terrestrial conditions	Successful
COST	The system shall cost < $4000	Successful ($3300)

7.3. Schedule and Budget Status

7.3.1 Spring Semester Schedule

The projected Spring development schedule with Progress Review milestones is depicted in figure 23. The primary goals for the Spring are the integration of new sensors, including cameras and an IMU, the redesign of both the chassis and attachment mechanism, implementation of software localization and path-planning, and the design and construction of a demo test environment. These tasks may be roughly partitioned between Software, Mechanical, and Electrical engineering tasks, though heavy crossover and collaboration is expected to occur.



Name	Begin date	End date
Software	1/13/14	4/3/14
Complete Simulation	1/13/14	1/29/14
Mapping Representation	1/30/14	2/12/14
Sensor Software Integration	1/20/14	1/31/14
Localization	2/3/14	3/7/14
Path Planning in Simulation	2/13/14	3/7/14
Waypoint Functionality	3/10/14	3/27/14
Path Planning in Demo Envi...	3/10/14	4/3/14
Mechanical	1/13/14	3/19/14
Prototype Redesign Strategi...	1/20/14	1/29/14
Sticky Foot mk.II Prototype	1/13/14	1/29/14
Final Prototype Design	1/30/14	2/12/14
Final Prototype Fabricated	2/13/14	2/28/14
Final Prototype Integrated	3/3/14	3/19/14
*Hardpoint Design	1/30/14	2/12/14
*Hardpoint Fabricated	2/13/14	2/20/14
*Hardpoint Integration	2/21/14	2/26/14
Electrical	1/13/14	2/12/14
Sensor Acquisition	1/13/14	1/17/14
Sensor Hardware Integration	1/20/14	2/12/14
Test Environment	1/13/14	4/3/14
Test Environment Designed	1/13/14	2/27/14
Test Environment Material A...	2/28/14	3/19/14
Test Environment Assembled	3/20/14	4/3/14
Flex Time to Catch Up	4/4/14	4/16/14
PR 7	1/30/14	1/30/14
PR 8	2/13/14	2/13/14
PR 9	2/27/14	2/27/14
PR 10	3/20/14	3/20/14
PR 11	4/3/14	4/3/14
PR 12	4/17/14	4/17/14

Figure 23. Planned development schedule for the spring semester

Development is mostly on schedule with the exception of adhesive foot design, which was more arduous than anticipated at the onset of the Fall semester. Waypoint functionality was erroneously forecasted to be implemented in the Fall semester, but was not feasible without elements such as localization and path-planning which are themselves scheduled for Spring implementation. Fortunately, some tasks are ahead of schedule, such as the development of a second prototype design to be built in the Spring, and the Rviz simulation/visualization, which was originally forecasted to be developed in the Spring but proved a valuable asset early for developing the gait generator in the absence of a completed prototype.

7.3.2 Current Budget Status

Overall, we are significantly under projected budget for the project, Our largest expenditure to date has been the purchase of our servos, which came to $791.31 (including linkages, calibration, and mounting hardware). To date we have spent only 55% of our projected total budget, and have not yet spent anything from our managerial overhead, which still remains for future risk mitigation. Detailed budget breakdowns may be seen in tables 11 to 13.

Table 11. Fall expenditures

Description	Prediction	Total	Difference
Electrical Components	$400	$267.64	$132.36
Chassis Hardware	$800	$582.69	$217.31
Servos + Hardware	$800	$791.31	$8.69
Radios	$200	$111.15	$88.85
Fall Total	**$2200**	**$1752.79**	**$447.21**

Table 12. Spring budget predictions

Description	Qty.	Cost	Total
Cameras	2	$150	$300
IMU	1	$150	$150
Spring demo environment	N/A	$100	$100
Misc. Hardware	N/A	$600	$600
Spring Total			**$1050**

Table 13. Total budget

Fall Expenditures	$1750
Spring Expenditures	$1050
Managerial Overhead	$500
Total	**$3300**

7.4. Risk Management

Table 14. Table of anticipated sources of risk

Risk	Associated Reqs	Description	Type	Cause	Likeli-hood	Conse-quences
Team Member Illness	Timely Project Completion	A team member is severely ill for a significant period	Schedule	Largely Chance	Somewhat Likely	Lost Man-Hours and Expertise
Insufficient Servo Torque	Mobility, Traversal	Unable to lift segments with given motors	Technical	Unexpectedly High Mass	Rather Likely	Reduced or No Mobility
Overweight Linear Actuators	Mobility, Traversal	COTS linear actuators too heavy	Technical	Servo Torque constraints	Extremely Likely	Reduced or No Mobility
Insufficient Adhesion	Attachment	Adhesive foot material provides Insufficient "Stick" to support robot	Technical	Robot Weight, Foot material properties	Somewhat Likely	Poor attachment on vertical or inverted surfaces
Actuator cost over-budget	Cost	Actuators and associated component cost exceed expectations	Cost	Budget constraints	Somewhat likely	Total project over-budget
ROS Package Incompati-bility	Technical	A ROS package we need is not compatible with our ROS version	Technical (Software)	ROS Developers	Likely	Limited software capability

Table 15. Risks with mitigation strategies and fall semester comments

Risks	Reduction Plan	Actions Planned	Expected Outcome	Comments
Team Member Illness	Redundant system knowledge between team members. Healthy living / Flu shots.	Primary and secondary assignees for all project tasks.	Minimal loss of expertise.	Endured team-mate sickness and kept project moving
Insufficient Servo Torque	Designed with Mid-range servos in mind, larger torques available if needed.	Weight budgeting and motor trade studies. Support additional gear/linkage reduction in design.	All torque needs met. Some time lost in redesign/shipping.	Chassis Flex ended up being the larger issue
Overweight linear actuators	Find workarounds for specialized actuators.	Retrofit and modify hobby servos to suit linear applications	Time lost, but lower weight actuators.	Built our own
Insufficient Adhesion	Hardpoint attachment strategy	Test feet with generous weight Develop hardpoint attachment backup plan	Smooth transition to hardpoint plan if necessary	Still a major issue
Actuator cost over-budget	Total project budget significantly less than class limit.	Cut into managerial overhead	Total budget does not exceed $4000	Aligned with budget
ROS Package Incompatibi-lity	Spec packages early	Attempt upgrade or write package ourselves	Lost Dev. Time	No Problem So far