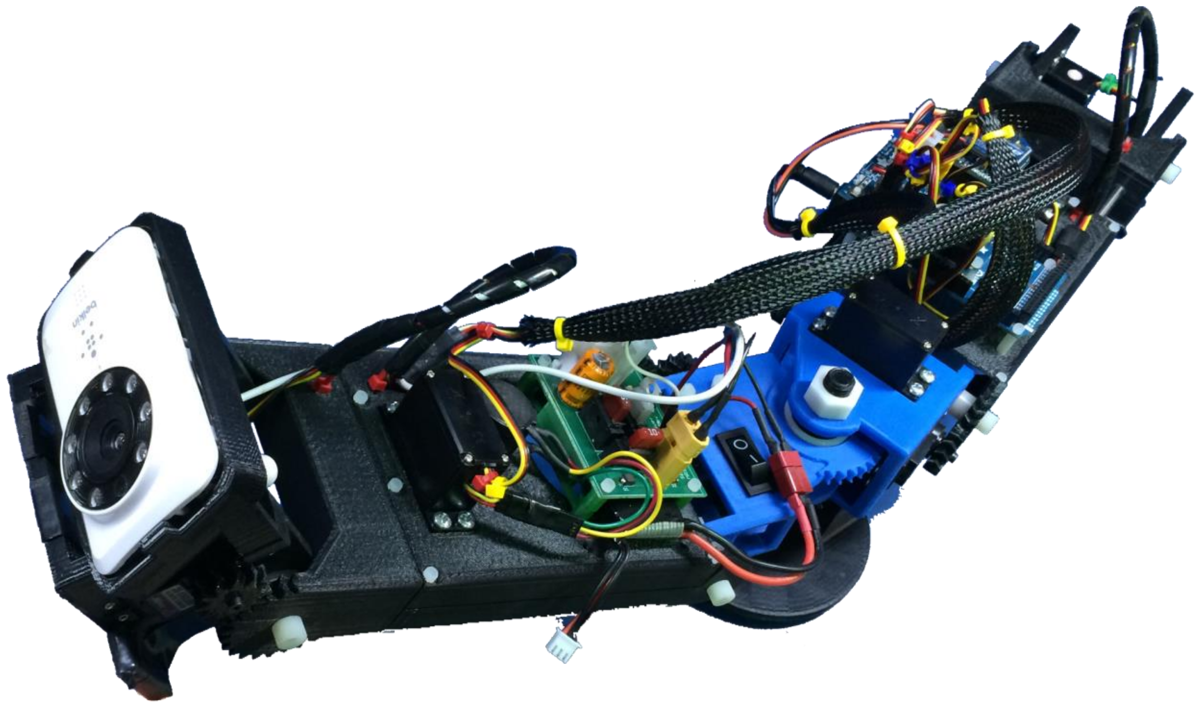


# Space Jockey

*A Mobile Robot Platform for Spacecraft Exterior Inspection & Maintenance*

Final Report - Team B

May 8th, 2014

*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 machines. 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

## 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. Space Shuttle Columbia 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 entirely supplant 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 portion of the hull surface 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.

### 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.

Our system is designed to be highly mobile so that it's capable of traversing the largest possible subset of a craft's surface. Using a camera, it will be able to gather information about its surroundings useful not only for localizing itself within the environment, but for inspecting the hull surface and presenting that information to an operator in such a way that he/she can understand and react to defects or flaws. By making our robot compact, light, and wireless, it can operate with minimal supporting infrastructure ad be amenable to the demanding costs of launch into space. Finally, with some elements of autonomy that allow it plan its own paths and execute them, we minimize the expertise and training necessary for operators, so that it will be useful to spacecraft crews without imposing substantial new requirements on mission planners that might otherwise cancel out the savings gained from fewer EVA missions.

## 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, traverses (2) the surface of the craft semi-autonomously by planning its own path to waypoints specified by the operator. While traversing, the robot inspects the surface as it passes over, and can stop for more detailed inspection or to perform repair tasks (3), and concludes its mission by returning (4) to its point of origin and stowing (5) until its next mission.

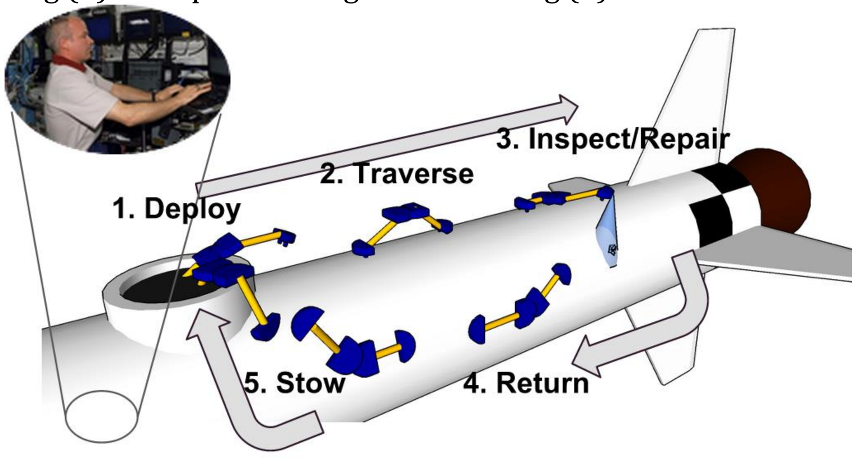


**Figure 2. Use Case of Space Jockey System**

## 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 Space jockey 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

| ID | Code | Description |
|---|---|---|
| 1 | ATTACH | Maintain attachment to the hull<br>TPM: Supports own weight on a 1G vertical surface |
| 2 | INSPECT | Inspect  the surface at a reasonable speed<br>TPM: 6 square meters per hour |
| 3 | FLAW DETECTION | Autonomous identification of surface flaws<br>TPM: 90% detection rate, with no more than one false positive over a 3 meter area |
| 4 | SENSE | Clear operator feedback of the current robot state and camera viewpoint |

#### 3.1.2 Non-Functional

| ID | Code | Description |
|---|---|---|
| 1 | TETHER | System must operate without a physical tether or power connection |
| 2 | EARTH | System can be validated in terrestrial conditions |
| 3 | MASS | Robot must be as light-weight as possible to reduce launch requirements<br>TPM: <10kg |
| 4 | SIZE | Robot must be able to be packed into minimal volume for efficient storage<br>TPM: 8 Cubesats (10cm cube) |
| 5 | BUDGET | Total system cost must not Exceed $4000 |

## 3.2 Desirable Requirements

### 3.2.1 Functional

| ID | 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 | PLANE | Plane Transitions (60-degree) |

### 3.2.2 Non-Functional

| ID | Code | Description |
|---|---|---|
| 2 | INFRA | Minimal Spacecraft Support Infrastructure |
| 3 | ATTACH2 | Maintain attachment to the hull<br>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.

### 4.1 Mobile Robot Subsystem

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 camera, and then transmits them to the command center over a radio link. The command center receives the signal from the robot and processes the images, warping according to localization knowledge and then mapping them onto a world map. This sensor map is then compared against previously known inspection data to detect any flaws. If significant deviations are discovered, the user is notified and the defect highlighted so that a repair plan may be generated. The user may also monitor the image feedback directly, both to intuit knowledge about the robot's position in addition to localization information and to personally inspect the hull. Even without our automatic flaw detection capability, this mobile eye in itself obviates can obviate certain EVA missions.

### 4.2 Command Center Subsystem

The operator may select a series of waypoints 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 transmit to the robot via the radio link, and the robot performs the desired maneuvers. Maneuvers can include both commands to move to a given location or to view a location, in which case the robot only moves as much as is necessary to bring the location into the lifted camera's field of vision. The robot's progress may be monitored through the command center, and its actions halted or modified as deemed necessary by the operator.

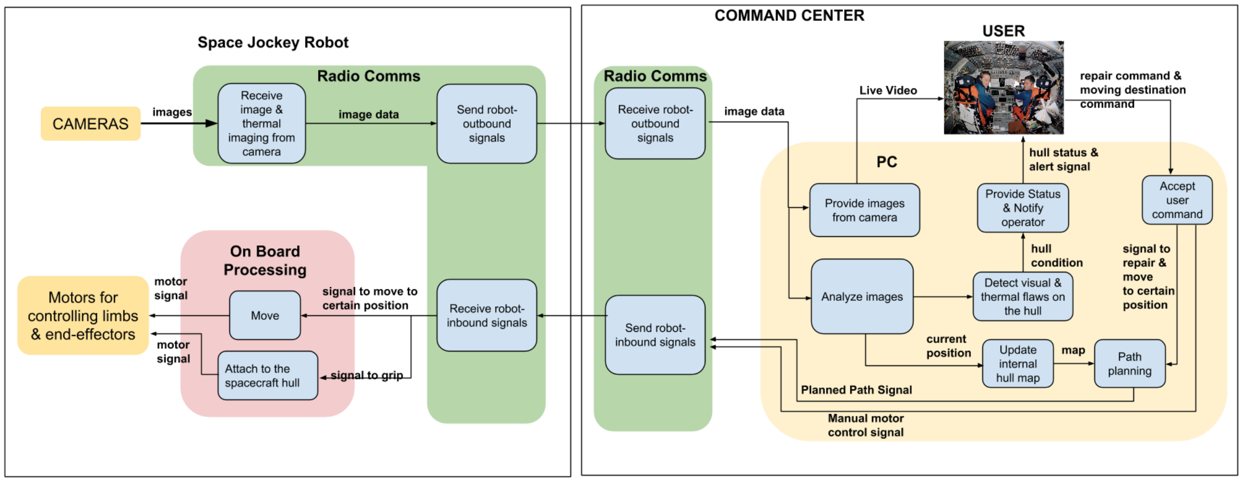

Space Jockey Robot
Radio Comms
CAMERAS
images
Receive image & thermal imaging from camera
image data
Send robot-outbound signals
On Board Processing
Motors for controlling limbs & end-effectors
motor signal
Move
signal to move to certain position
Receive robot-inbound signals
motor signal
Attach to the spacecraft hull
signal to grip
COMMAND CENTER
USER
Radio Comms
Receive robot-outbound signals
image data
Live Video
repair command & moving destination command
PC
hull status & alert signal
Provide images from camera
Provide Status & Notify operator
Accept user command
hull condition
signal to repair & move to certain position
Analyze images
Detect visual & thermal flaws on the hull
Send robot-inbound signals
current position
Update internal hull map
map
Path planning
Planned Path Signal
Manual motor control signal

**Figure 3. Functional Architecture**

## 5 System Level Trade Studies

### 5.1 Mobility Study

The chief challenge to be tackled in our robot design was the question of mobility. Providing a reliable, stable platform for tooling and inspection equipment in a zero gravity and hard vacuum is a difficult problem. Many approaches to this have been explored in the past by the space industry, and recent advances in dry adhesive technology have opened up new possibilities. While we were initially focused on exploring this technology, it will later be explained why we ultimately chose another attachment mechanism using more proven and available means, magnets. Assuming a ferromagnetic surface, however, the design options for adhesive pads translate easily into one with magnetic attachment.

### 5.1.1 Mobility Design Options

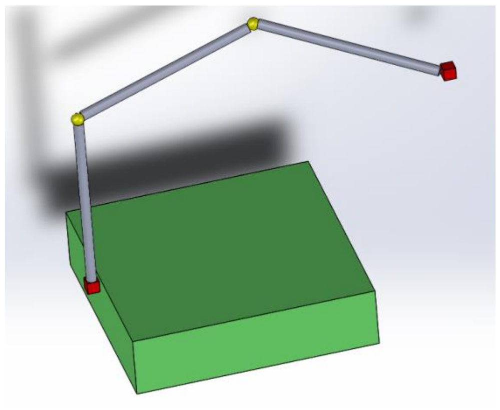


**Figure 4. Dextrous arm with Hardpoints**

This design is very similar to the Canadarm 2[1] onboard the ISS. A large remote manipulator is fitted to the exterior of the craft and may "walk" arm-over-arm about the exterior of the craft by grasping hardpoints.

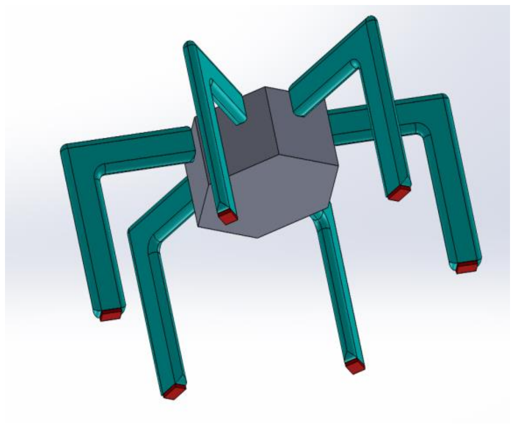


**Figure 5. Multi-legged robot with Adhesive Feet**

This design is similar to the AWIMR[2] project developed at JPL and relies on new advancements in conformant materials or dry adhesives to attach to the craft exterior.

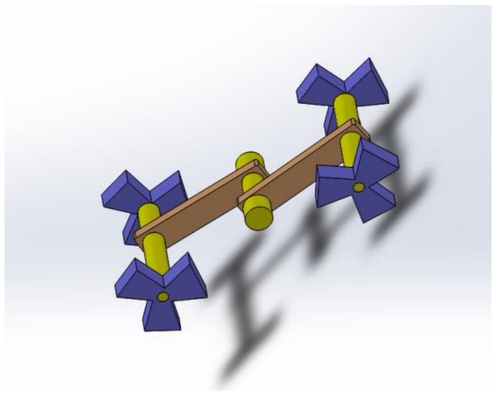


**Figure 6. Wheeled Robot with Adhesive pads**

This design is similar in form to the Waalbot, developed at CMU[3], this option uses wheels as the basis for mobility. The wheels are composed of discrete pads of dry-adhesive material to adhere to the surface, and employ passive mechanisms to "peel" pads away from the surface.

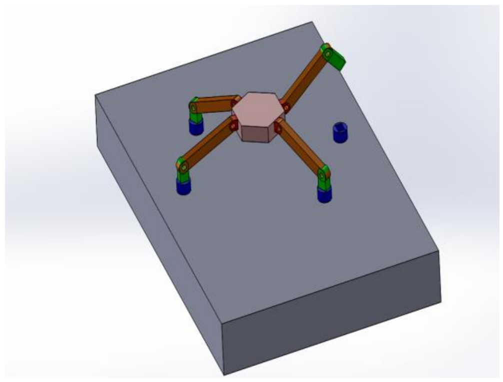


**Figure 7. Multi-legged Robot with Hardpoints**

Combining elements of the first and second options, this robot navigates the hull surface using external hardpoints to anchor itself. This approach has much in common with the discontinued Lemur IIb[4] design developed by the JPL in 2006.

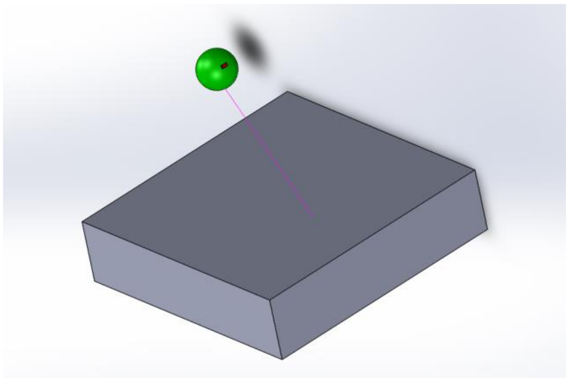


**Figure 8. Tethered Robot**

In this concept, a robot is attached to the craft using a tether and surveying the craft from a distance, held by centripetal forces. Some research has already been done into tethered Satellite deployment[5] and we could leverage that work for inspection tasks.

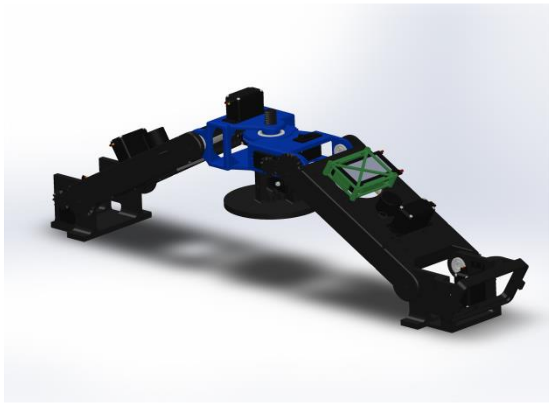


**Figure 9. Inchworm Robot**

A robot is attached to the craft using adhesive or magnetic feet. The robot's unique structure makes it able to do complex operations and maneuvers, while remaining mechanically simple.

### 5.1.2 Selection Criteria

| Criteria | Weight | Description |
|---|---|---|
| Mobility | 20% | The speed and ease with which each design option is able to reach its inspection goals. |
| Fuel / Power requirements | 5% | Keeping the overall robot as light and as simple as possible is a virtue in the design of space systems. |
| Safety systems / Fallback | 10% | The relative danger of the robot damaging the spacecraft or being lost in case of a control failure. |
| Infrastructure Requirements | 10% | Requirements for handholds, power stations, tethers, or other devices to be affixed to the spacecraft hull increases the cost of deployment of our system. |
| Platform stability | 10% | As one of our desired tasks for our robot long term is tool usage, the ability of the robot to provide a rigid platform for manipulation tasks is weighed in this metric. |
| Complexity | 15% | The relative complexity of the design task must be weighed to guarantee feasibility within the time constraints. |
| Testing Feasibility | 20% | Since we do not have access to a zero-g testing facility, the feasibility of system verification is evaluated by this metric. |
| Proven Technology | 10% | This metric weighs the reliance of each design on untested or not readily available technology. Reliance on difficult to acquire parts or pieces could jeopardize the success of the project. |

### 5.1.3 Study Scoring Matrix

| Dexterous Arm | Legged Hardpoint | Legged Adhesive feet | Wheeled Adhesive Pads | Tethered | Inchworm Robot | Criteria | Weight |
|---|---|---|---|---|---|---|---|
| 6 | 6 | 8 | 7 | 3 | 9 | Mobility | 20% |
| 10 | 8 | 6 | 7 | 10 | 6 | Power | 5% |
| 8 | 8 | 6 | 6 | 1 | 6 | Safety | 10% |
| 4 | 2 | 8 | 8 | 4 | 8 | Infra-structure | 10% |
| 8 | 10 | 8 | 6 | 0 | 8 | Stability | 10% |
| 7 | 7 | 8 | 10 | 3 | 9 | Complexity | 15% |
| 8 | 10 | 10 | 10 | 1 | 10 | Test Feasibility | 20% |
| 10 | 10 | 3 | 3 | 5 | 4 | Proven Technology | 10% |
| 7.35 | 7.65 | 7.6 | 7.55 | 2.75 | 8.05 | Totals | 100% |

### 5.1.4 Results

Based on the results of our trade study, we chose to pursue the inchworm robot. This design was not only relatively simple compared to, say, a six-legged crawling robot, but it did so while maintaining a very high degree of mobility, able to traverse angled surfaces and turn on a dime. Additionally, though not mentioned in the trade study, it was to our knowledge a completely novel mobility system which made it an exciting prospect. However, a key issue that is alluded to in the "Proven Technology" criterion is that we were unsure of just how applicable dry-adhesive surfaces would be to common spacecraft surface materials.

## 5.2 Attachment Mechanism Rationale

In deciding on an attachment mechanism, the conflict between our desired functionality and capabilities for testing was a large factor. The most ideal solution for a system operating on actual spacecraft in orbit would be a dry-adhesive material able to adhere to a variety of surfaces without causing damage or leaving residue. However, to test in Earth gravity, the holding force of whatever attachment mechanism was chosen had to be much stronger than necessary for the targeted application, and so this battle with gravity became a defining struggle in development. In practice, we were unable to obtain a suitable, high-quality material capable of withstanding gravity, and eventually had to abandon this route in favor of magnets.

Another alternative, the use of hardpoints which the robot segments could grasp or otherwise mechanically attach to, was avoided due to the complexity and difficulty of the resulting peg-in-hole problem. Magnets were less favored given that most spacecraft surfaces are not ferromagnetic, typically aluminum or insulating materials. However, they were nonetheless adequate for proving the mobility system in Earth gravity, and ultimately offered the best balance between holding force, simplicity, and availability.

## 6. 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 GUI attached to ROS. The full physical architecture is shown in Figure 10.

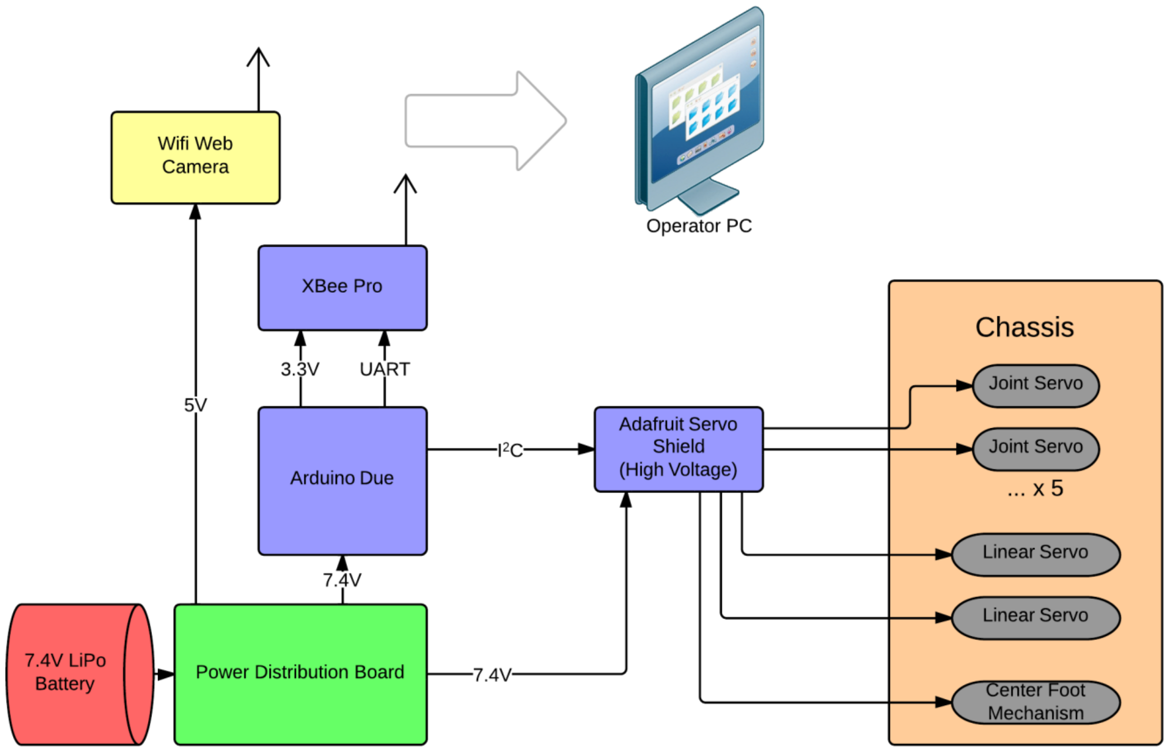


**Figure 10. Physical Architecture Diagram**

The robot's onboard electronics are all powered by a 7.4V LiPo battery, whose voltage is regulated and distributed by our custom power distribution board. Our onboard processor, an Arduino Due, communicates with operator PC through serial radio communications, and interfaces with the servos on the robot through an Adafruit PWM controller shield connected via an I2C bus. Inspection data is transmitted directly to the operator's PC from a Wi-Fi enabled Web camera attached to the robot. All control signals and planning tasks are then performed onboard the operator PC.

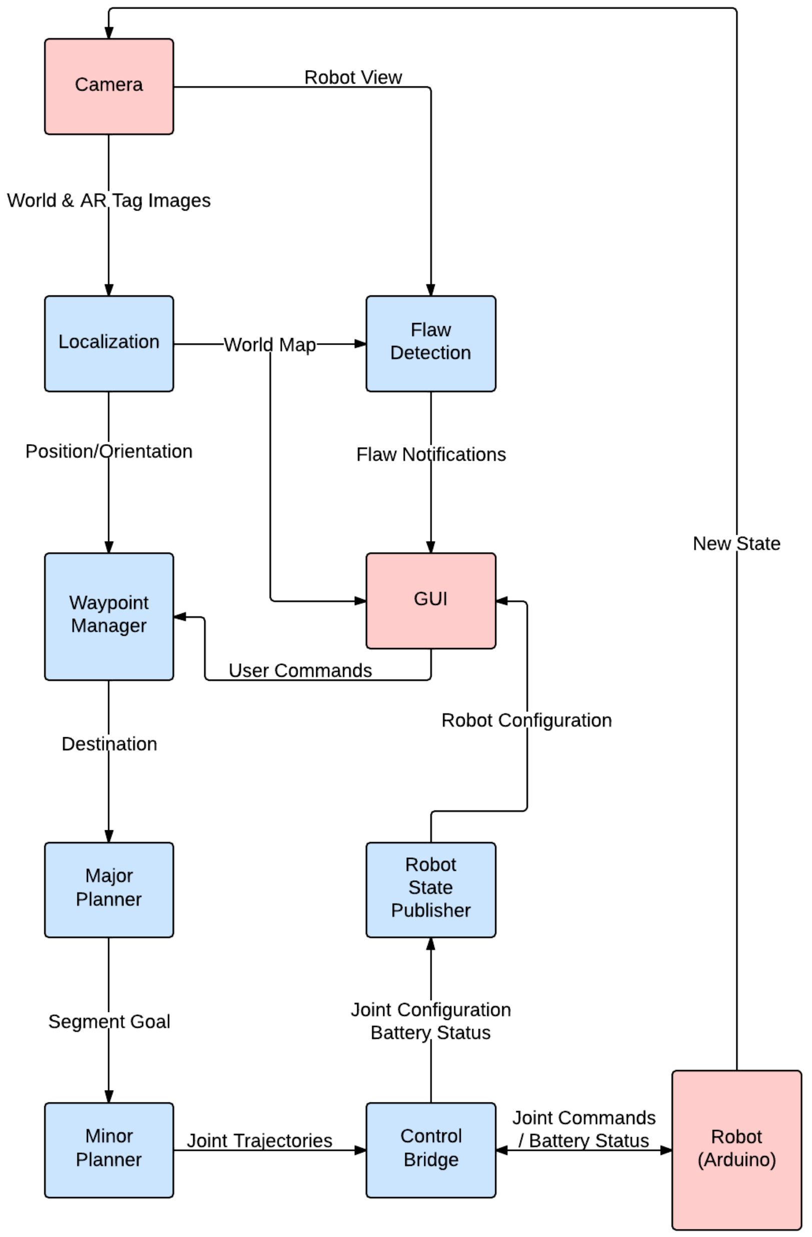


**Figure 11. Space Jockey Software Architecture**

The information flow within the control system is depicted in figure 11. Images of the world from the rover's perspective are gathered by the camera and transmitted to the command console, where they provide the operator with a robot's-eye-view of the surface as well as views of AR tags which enable localization. Using the knowledge of position and orientation provided by the localization module, images from the robot may be warped and superimposed on the "clean" map of the world. By comparing this live map with the "clean" map, a flaw detection module identifies significant differences and alerts the user of probable defects, highlighting them with a bounding box on the GUI.

To command the robot to move, the operator clicks to create a waypoint on the map, or drags to select a region of inspection, which is then populated with a grid of waypoints. The command console will then generate a path that allows the robot to view each waypoint. The "Major Planner" module decides the series of segment motions that will bring the robot to the point, and delegates the task of generating precise joint trajectories of each segment to the "Minor Planner".

Joint trajectories generated by the minor planner are then scheduled and dispatched to the robot and the robot state publisher. The speed of dispatch is governed by known maximum joint velocities of the robot. As commands are sent, the updated joint configuration is used to update the kinematic model of the robot, which is displayed in the Operator GUI and in an optional RViz window as well.

## 7 System description and evaluation

### 7.1 Subsystem breakdown

The system's subsystem breakdown is shown in  figure 12. Note that command center subsystems constitute software processes implemented in ROS, while those of the robot are primarily hardware components. Essential mobility was the focus of the fall semester, while during the spring we implemented peripherals (cameras, IMU) and elements of autonomy (localization, planning).

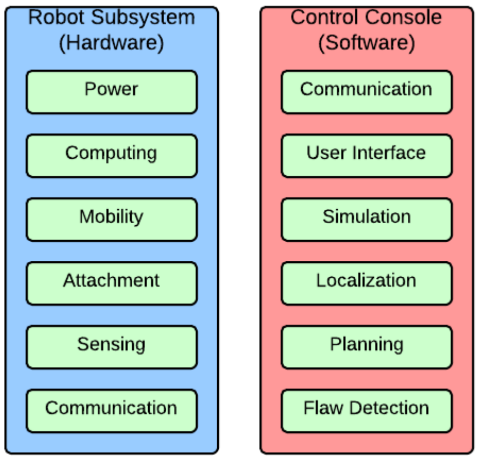


**Figure 12. Subsystem Breakdown**

## 7.2 Robot Subsystems

### 7.2.1 Power

The robot is powered by a 7.4V hobby LiPo battery. All power is routed through the power distribution board, which provides two fused 7.4V channels for the Arduino and servo control boards, and one regulated 5V channel for the inspection camera. The completed board is shown in figure 13 and its PCB layout in figure 14.

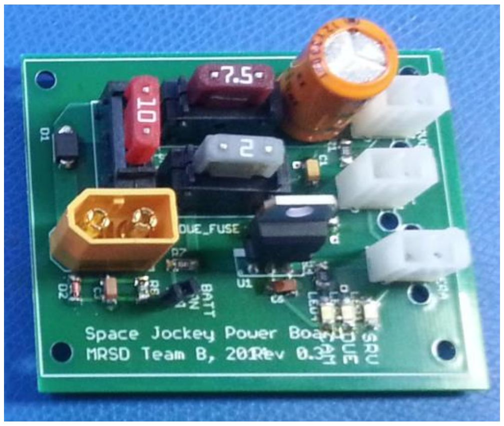


**Figure 13. Completed PDM Board**

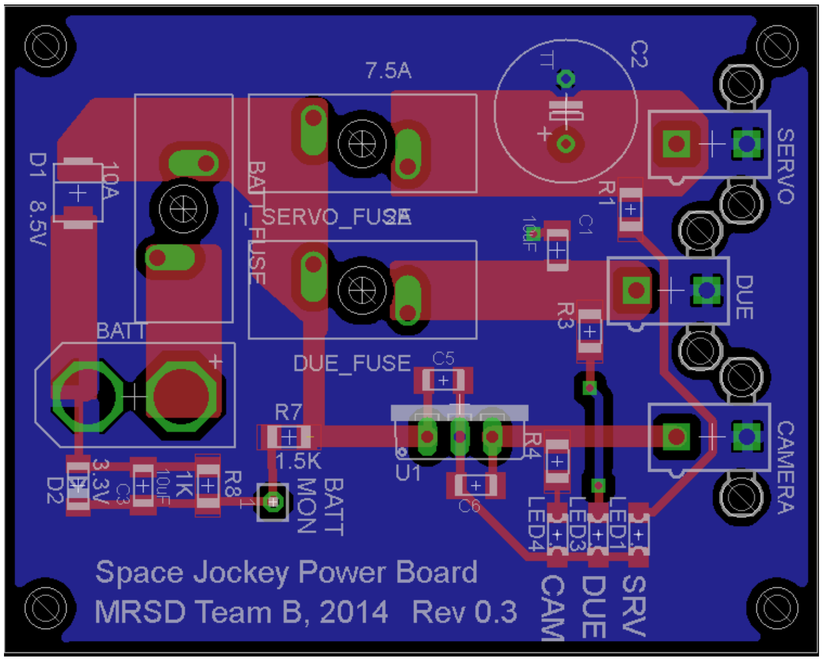


**Figure 14. Power Board Layout**

### 7.2.2 Computing

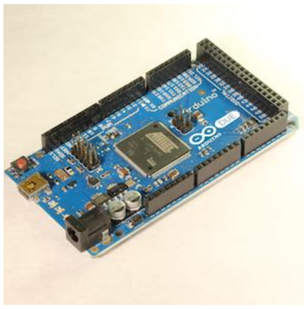


**Figure 15. Arduino Due**

The microcontroller for our chassis is an Arduino Due (Figure 15). The system is controlled by the Arduino and communicated to a laptop through serial radio communication 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.

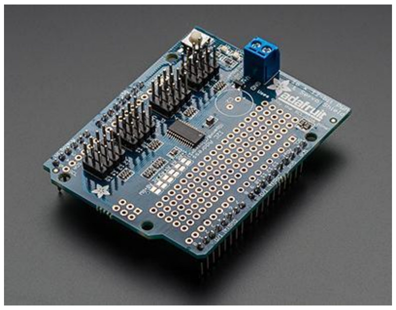


**Figure 16. Adafruit PWM Driver**

The interface between the microcontroller and servo motors is the Adafruit Industries 16 Channel Servo/PWM controller shield (Figure 16). The shield's 12-bit interface allows 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. To minimize control system frequency drift due to temperature changes or clock instability, one PWM channel is fed back into the Arduino, allowing us to precisely calibrate our servo board every time the robot boots up.

### 7.2.3 Mobility

Figure 17 shows the design of the robot's chassis in Solidworks. Because we were unable 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 minimize the total mass of the robot. The final robot's mass, including all motors and electronics, is about 1.65kg.

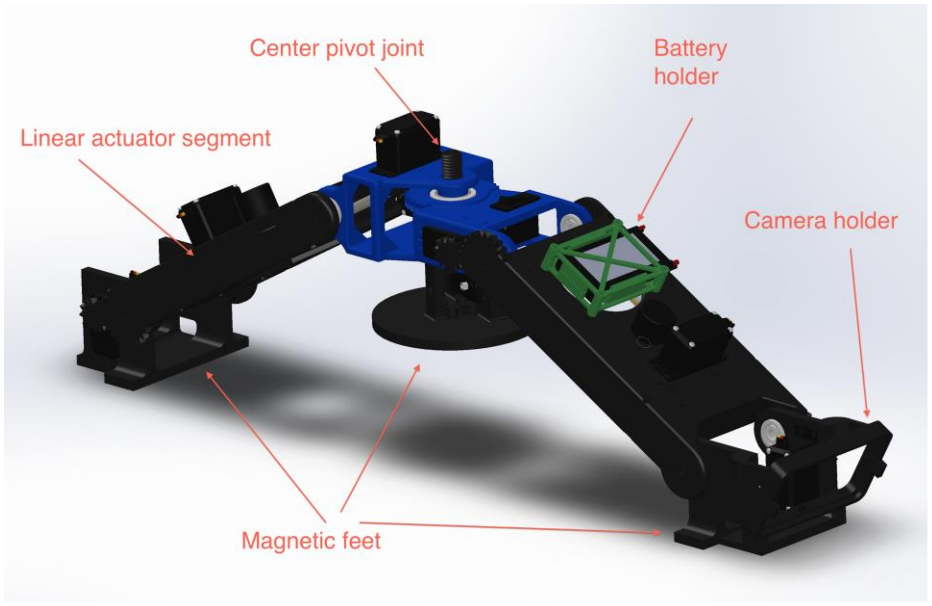


**Figure 17. Solidworks Model of Space Jockey Chassis**

The chassis was fabricated using 3D printed ABS parts printed on the MRSD Makerbot. This fabrication method allowed rapid redesign and prototype iteration turnaround. Each major chassis component required 4 hours of time to print on average. This rapid turnaround, combined with the part complexity we were able to generate using the 3D printer, allowed us to develop a very lightweight, well-integrated chassis. Although we did have some issues with breakage and layer separation of the ABS parts, the design flexibility of this fabrication method was unmatched. The original prototype had been made with laser-cut MDF, and switching to ABS allowed us to not only implement a much tighter form factor, but cut the robot's net weight by 60%.

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 18). 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 18. Hitec HS-5585MH Servo**

For the linear actuators for which less torque is 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 19) 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 connection to the command center.

**Figure 19. HPP-21 Servo Programmer**

### 7.2.4 Attachment

Each of the three robot segments has a "foot" mounted to it which is responsible for maintaining the attachment of the robot to the surface. For the center segment, the foot is a separate assembly topped with a screw which is guided through the central pivot joint (See Figure 20). For the two end segments, the foot surface is integrated directly into the 3D printed segment (See Figure 21).

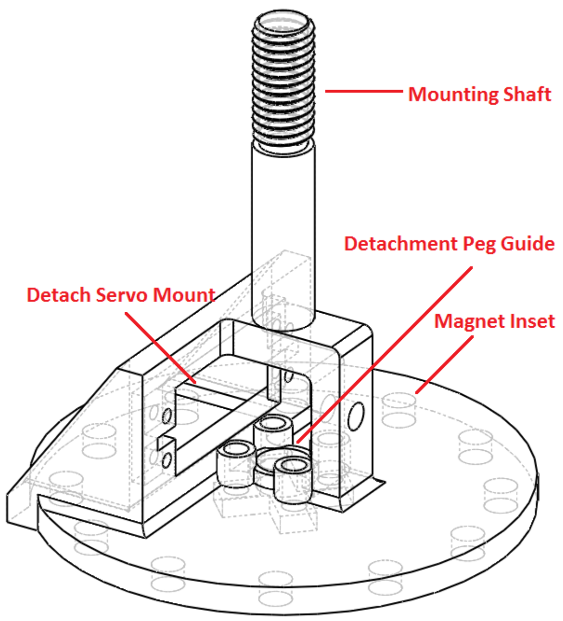


**Figure 20. Center Segment Foot Design**

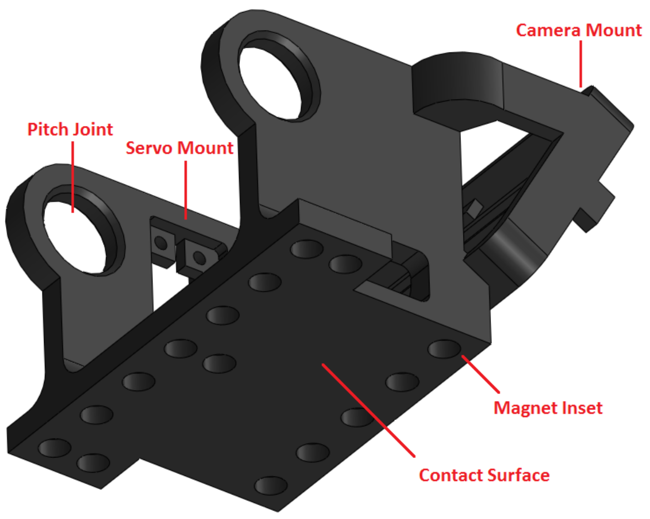


**Figure 21. Front End Segment with Integrated Foot Mechanism**

Attachment to the inspection surface is accomplished using an array of magnets inset within the contact surface of each foot. Each foot is 3D printed and two magnets are secured in each inset using epoxy. To prevent slip between a foot and the surface, a layer of high-friction, off-the-shelf grip tape is applied to the contact surface after the magnets have been installed. The number of magnets in each foot is intentionally more than necessary to secure it against most surfaces, and additional layers of tape may then be used to increase the distance between the surface and the magnets and thus adjust the total attachment force.

Detachment is accomplished differently for the front and end segments. End segments can be independently pitched forward or backward prior to lifting, levering the foot about one of its edges. This forces enough magnets away from the surface that the entire foot can then easily be lifted. To place the end segment again, the foot needs only to be leveled parallel to the surface and lowered to meet it. In the case of the center segment foot, an actuated solid plunger is inset in the center of the foot and can be extended to push the foot away from the surface as other actuated joints lift it simultaneously. Placing the center segment is a matter of retracting the detachment plunger and lowering the foot to again meet the surface.

### 7.2.5 Sensing

Our inspection data is collected using a physically modified Belkin NetCam Wi-Fi enabled camera (figure 22) integrated with the robot frame. Using this camera allowed us to simplify the onboard processing system of our robot, as image data is transmitted directly to the operator PC, without the need for the onboard processor to do any image processing. The camera was attached to campus Wi-Fi, and

**Figure 22. Belkin NetCam**

could be accessed from anywhere. The use of a webcam also simplified testing, as several people could connect to the camera stream at the same time, and view or troubleshoot their software components separately.

### 7.2.6 Communication

Serial communication of commands and robot status information is transmitted between the control console and the robot using XBee Pro 60mW radio modules (figure 23), which implement the IEEE 802.15.4 standard. These simple devices act as a drop in replacement for the USB cable on the Arduino, allowing our ROS software implementation to be completely agnostic of the transport medium.

**Figure 23. XBee Pro Wireless Module**

## 7.3 Command Console Subsystems

The software ecosystem we used for development of our code was based in ROS. We implemented much of our custom software ourselves using Python  and C++, but still tried to use community-made packages wherever possible.

### 7.3.1 Communication

Our primary communications with the robot is provided by the rosserial-arduino[6] package, this allows us to publish and subscribe to ROS topics natively using the onboard control electronics. Battery state messages, IMU information (which is currently removed from the robot, because it was unused), and joint control signals are transmitted via rosserial-python, which handles all the serial hardware implementation, to our ROS environment. In addition, to simplify hardware calibration, all of our robot's configuration was stored in rosparam yaml files, and then joint angles were transformed into servo control pulse lengths (in ms) by the control bridge node, which allowed us to easily calibrate our robot's joint ranges, servo channel wiring, or battery calibration data without having to recompile and re-flash our Arduino code.

Inspection data was passed into ROS by our custom ipcam[7] package, which can subscribe to an MJPEG webcam feed by URL, and then translate that into a ROS image topic, to be parsed and processed by our localization and flaw detection nodes.

### 7.3.2 User Interface

The graphical user interface (GUI) was developed using Python and OpenCV. It shows the user a dim image of the clean world map, the current position of the robot's feet (blue dots), and the current battery status. Users may issue commands by either right-clicking, which adds a movement waypoint command (red circles), or clicking and dragging to select an inspection region of interest (green circles). A screenshot of the GUI may be seen in figure 24.

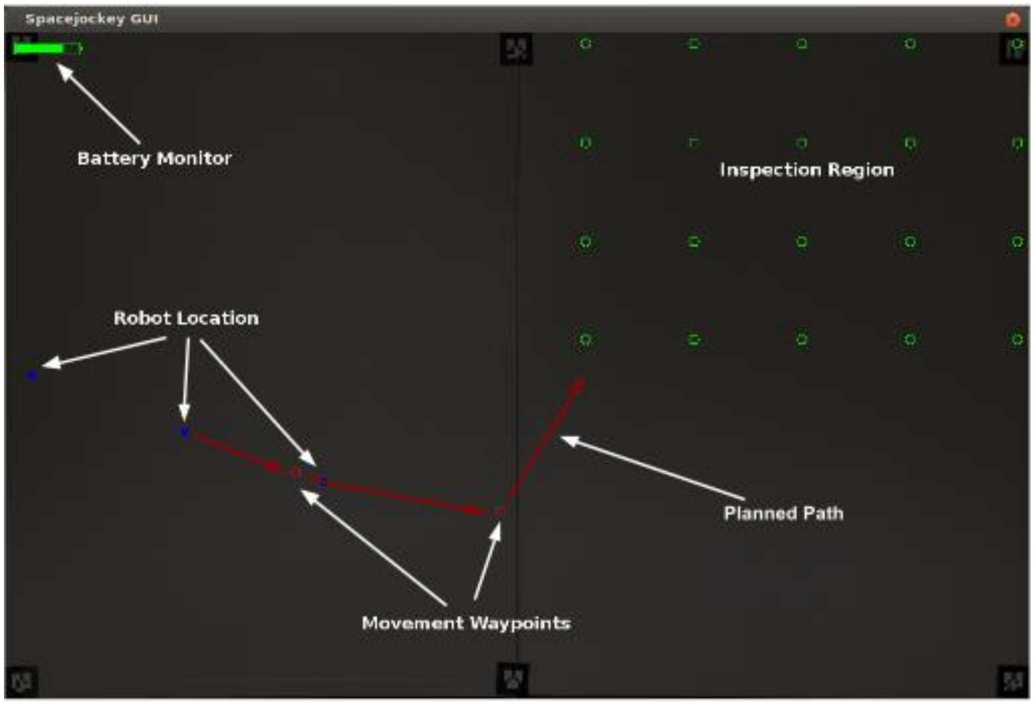


**Figure 24. Screenshot of the GUI, showing basic features.**

Once commands have been issued, the robot will autonomously execute the given instructions. As inspection takes place, the updated inspection data is overlaid on the GUI, and any detected flaws are clearly marked with red bounding box, as seen in figure 25.

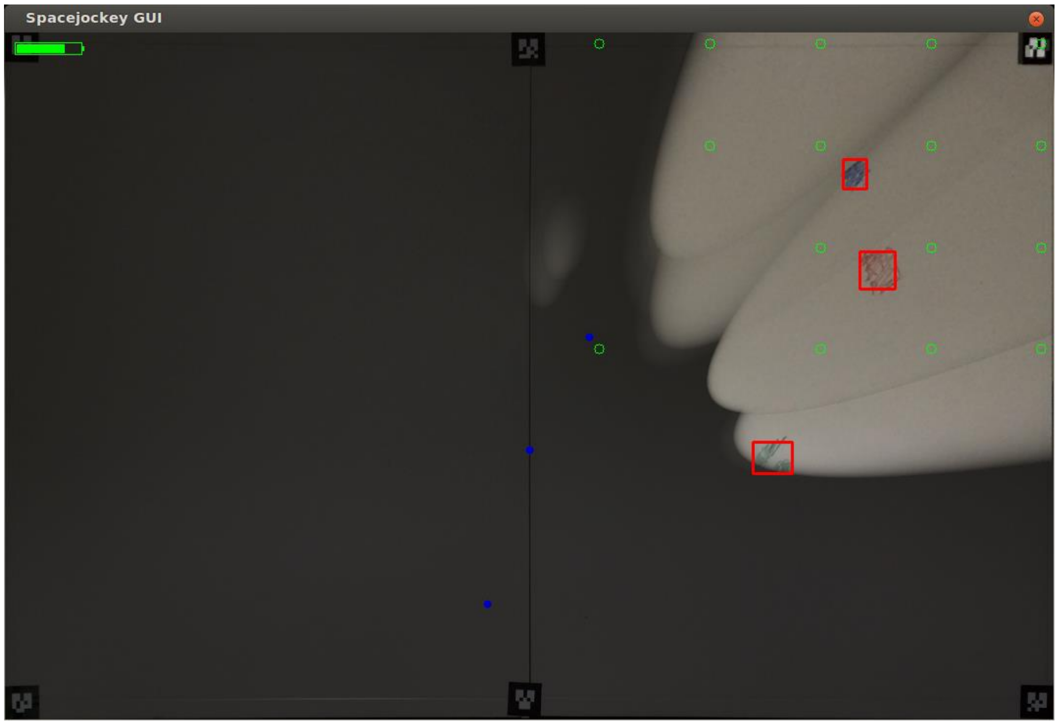
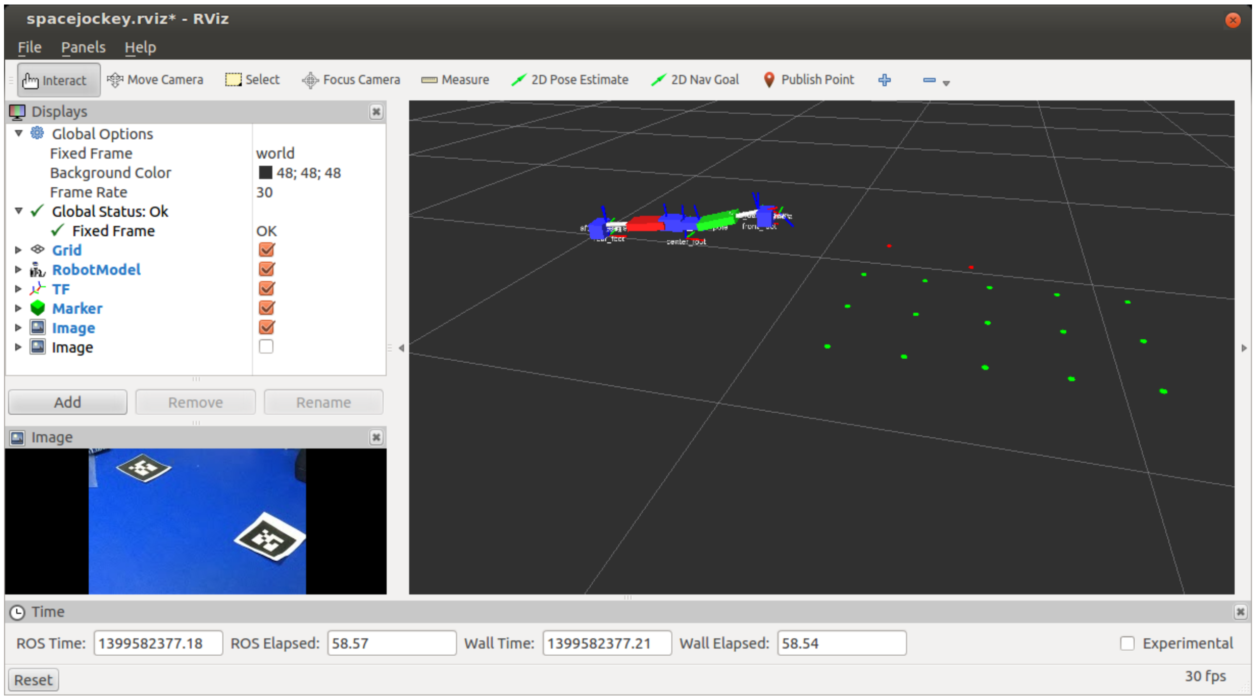


**Figure 25. GUI screenshot showing partial surface inspection and detected flaws. (using simulated camera data)**

In addition to camera data, the GUI also subscribes to the battery state topic, and vizualization_msgs, which is used to render current waypoints (updated from the major planner) and flaw notifications (from the flaw detection node).

### 7.3.3 Simulation

In addition to the main GUI, additional rendering of the world state can be seen using Rviz. This allows us to easily view the robot's 3D state, raw camera data, coordinate frame locations, and AR tag markers, which is helpful for debugging the robot state when things go wrong. However for clarity and simplicity during normal operations, these features were intentionally not included in the primary command GUI, and Rviz is not necessary to run the robot. The robot's 3D model is generated from a URDF definition file, and is updated by subscribing to the robot state publisher. Similarly to the GUI, any flaws or waypoint visualization messages are also rendered in RViz, providing an easy viewpoint. In addition, Rviz will also display the pose and ID of any detected AR tags, which helps in debugging the localization system of the robot. A screenshot of the Rviz window may be seen in figure 26.



Global Options
Fixed Frame: world
Background Color: 48; 48; 48
Frame Rate: 30
Global Status: Ok
Fixed Frame: OK

ROS Time: 1399582377.18 ROS Elapsed: 58.57 Wall Time: 1399582377.21 Wall Elapsed: 58.54

**Figure 26. RViz screenshot showing example camera imagery and robot pose.**

### 7.3.4 Localization

In the Space Jockey system, localization is accomplished by attaching fiducial markers (AR tags) to the inspection surface at known locations. These tags are identified from camera imagery by the ar_track_alvar package[8]. The localization node uses AR tag pose information provided by this package, back solves to find the robot's pose in the world frame, and then performs a weighted interpolation of this information with the robot's last known position, based on the robot's certainty of its current joint configuration. However, as of the SVE, this process was producing poor estimates of the robot pose, causing the rear foot of the robot to try to push through the ground surface, and popping the robot off of the wall, so this feature is currently disabled.

### 7.3.5 Planning

The planner pipeline is split into several different planners. Once waypoints have been specified in the GUI, this information is passed into the major planner, which maintains a priority queue of waypoints, and breaks down the designated route into "major actions" which are a sequence of foot movements, pauses, and inspection actions necessary to achieve the desired goal. The major planner has a complex decision tree within it, which makes sure that none of the planned actions violate the robot's joint limits, the basic pseudocode follows:

```
If the robot's feet are oriented at the target:
   if the target is within view range:
      point the camera at the target and pause 3 seconds
   else
      take a step towards the target with the next movable foot
else:
   turn until both feet are oriented at the target, respecting center joint limits
```

Once these major actions have been planned, the minor planner queries the robot's current state and uses our inverse kinematics solver to plan a joint trajectory from the current state to the target position. Because of the simplicity of our kinematic design and the cumbersome implementations of ROS' KDL and IKfast implementations, we chose to implement our own geometric solver for our robot. Control of our detachment mechanisms is also handled by our IK library using an optional parameter. Once joint trajectories have been planned, the minor planner then interpolates these over the target path, limited by the joint velocities specified in our robot's configuration files. The joint state is then output to the robot via our control bridge, and is updated in the GUI and Rviz via the robot state publisher node, which is ROS' built-in forward kinematics solver.

### 7.3.6 Flaw Detection

The first step in the flaw detection pipeline is to transform the raw camera input images into the world frame. Camera images are pulled from the ROS topic using the CV bridge[9] package, and then warped onto the world frame XY plane using OpenCV. In this case, using feature detection methods to get the transform matrix for warping is not suitable because the majority of the world map is featureless. Instead, we are using an affine transformation matrix K, combined with the camera position matrices R and T, which are pulled from the robot state, to derive a transform matrix H to warp the camera input image to world map. K is a 3X3 camera matrix discovered from camera calibration. R is a 3x3 rotation matrix of the camera's orientation relative to the world frame. However, we can delete the third column of R since Z values for all the points on the world frame XY plane are zero. So R becomes a 3X2 matrix for our situation. T is 3X1 translation vector describing the camera's position relative to the world frame. Then the warping equation is:

$$[u, v, 1]' \equiv K \cdot [R|T] \cdot [X, Y, 1]'$$

Once warped, these inspection images are accumulated into a 'dirty map' image, and transmitted to the GUI and flaw detection nodes using ROS.

The flaw detection node compares the clean map image of the original world state, with the dirty map derived from the camera data. Both images are passed through a Gaussian blur filter, and then subtracted from one another in the HSV color space. This difference image is weighted and thresholded using the hue and saturation of the differences. The pixel "value" is not used, as it is the most susceptible to changes in lighting. Once differences are found, a K-means cluster method is used to group these points into flaws. The bounding box of each flaw is then calculated, and passed back to the operator GUI as a visualization marker.

## 7.4 Modeling, analysis, and testing

### 7.4.1 Magnetic foot attachment testing

The number of magnets in each foot was determined through trial and error. Test rigs were made with a small number of magnets which were then tested for the normal and shear forces they could resist, and using those results we were able to predict roughly how many magnets would be necessary for each foot. Assuming that at any given time at least two feet would be attached, it was necessary that each foot be capable of supporting one half of the robot's weight.

This rough empirical approach resulted in a configuration of magnets which could indeed support the robot, but not under all circumstances. When an end segment was significantly cantilevered away from the surface, the additional force was sufficient to pull the center foot away from the surface or begin sliding. In addition, sudden motions by the robot would also cause slipping and detaching incidents, making it clear that the magnets we'd used to overcome static forces were insufficient against dynamic ones. Magnets and high-friction tape were added in subsequent iterations until they were more than capable of supporting the robot under a variety of configurations and motions.

### 7.4.2 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 27.

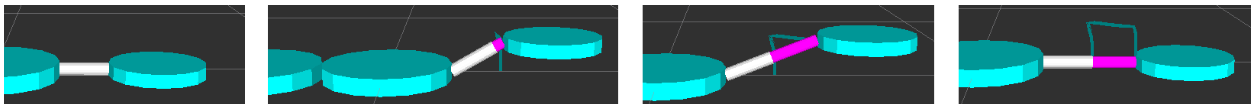


**Figure 27. The robot's front segment performing a square step forward.**

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.

## 7.5 System Performance

### 7.5.1 Spring-semester Capability Milestones

The capability milestones for the Spring Progress Reviews were laid out to space development of key functionalities over the months of January through March. In Table 1 below, the tests performed at each milestone are listed with a PASS/FAIL result.

**Table 1: Progress Review Dates and Tasks to be completed**

| Test Date | Progress Review | Tests Performed | Results |
|---|---|---|---|
| Feb 13 | # 8 | Adhesive attachment weight test (2kg)<br>Demo of redesigned linear and rotary actuators<br>April tag localization test | FAIL<br>PASS<br>FAIL |
| Feb 27 | # 9 | Pose estimation<br>Magnetic attachment weight test (2kg) | FAIL<br>PASS |
| Mar 20 | # 10 | New prototype horizontal walk<br>Simulated path planning<br>Image Comparison | PASS<br>PASS<br>PASS |
| Apr 3 | # 11 | Vertical surface walk<br>IMU Integration<br>Camera Integration | FAIL<br>PASS<br>PASS |
| Apr 15 | #12 | Localization system | FAIL |

|  |  |  |  |
|---|---|---|---|
|  |  | Vertical surface walk<br>Flaw detection | FAIL<br>FAIL |
| Apr 24 | Spring Validation Test | See 7.7.2 |  |
| Apr 30 | SVE Redux | See 7.7.2 |  |

### 7.5.2 Spring Validation Experiment

**Test Conditions**:

All of the tests and demo were held in NSH Level B. Required equipment consists of the robot command center, a metal surface with area of 3 square meters, dry-erase markers, magnetic AR tags, and the battery-powered robot. The metal surface was marked with red, green, and blue markers at arbitrary locations to signify flaws, and AR tags were affixed along the outer edge to bound the workspace and provide points of reference for localization. An assembled test rig can be seen in figure xx.

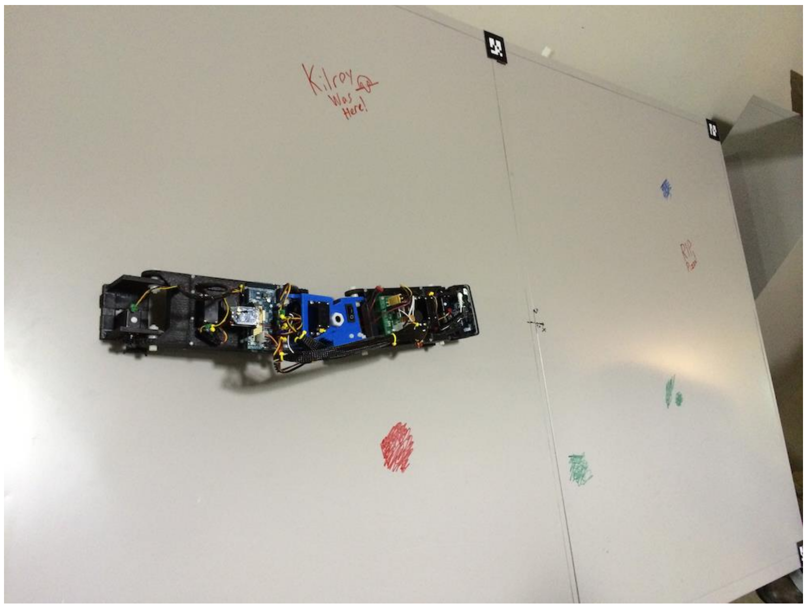


**Figure 28. Test area during spring validation experiment.**

**Equipment:**

Space Jockey Robot
Ubuntu PC with ROS and Control GUI installed
Clean 1m x 3m vertical surface with AR tags affixed in known locations
Misc. colors of dry-erase markers to create defects

**Procedure**

1. Mark surface with a random number ( > 5) of dry-erase marker drawn defects.
2. Power on robot and operator interface, connect to camera feed.
3. Place robot on the surface in a random location.
   - Handler remains nearby, spotting the robot in case of falls.
4. Operator initializes localization routine.
5. Operator selects regions of interest for inspection routine and observes robot operations

**Success conditions**

1. Robot successfully localizes based on AR tag information
2. Robot navigates and inspects vertical test environment, notifying operator of defects
   - Each drawn mark signifies a defect
   - Notifications are displayed as a bounding box around the flaw in the GUI
3. "90% success rate" in identifying flaws:
   - At least 9 of 10 flaws are identified and properly bounded
   - At least 9 of 10 identified flaws correspond to real flaws (not false positives)
4. Robot must not detach and fall from the surface at any point during the test.
5. The robot must complete the inspection routine in no more than 30 minutes

### 7.5.3 Spring Validation Results

✘90% success rate in identifying defects

- Current flaw detection implementation is overzealous and sensitive to lighting conditions, producing many false positives.

✘The localization routine was not functional at the SVE

- Testing was performed manual position updates.

✔The robot must not detach from surface and fall at any point during testing.

✔The robot must inspect at a rate of 6 meters$^2$ per hour.

✔The robot must perform autonomously when given an inspection command

✔The robot has must weigh less than 10 Kg  (final weight: 1.65 Kg)

✔The robot must operate wirelessly.

This semester, we were able to meet most of mandatory system requirements. During testing, the robot could support its own weight using a magnetic foot in Earth gravity and traverse the vertical surface. It met our size and weight goals and was capable of inspecting the 3 square-meter surface in about 20 minutes. Unfortunately, our flaw detection implementation was overzealous, and returned many false positives due to lighting, image projection, and shadowing issues. Also, our localization routine had several critical bugs that prevented it from being used at the SVE.

## 7.6 Strengths and Weaknesses

Testing and integration trials have made clear the strengths and weaknesses of the system as developed so far, summarized in Table 2.

Power and computing performance are both not ideal, but aren't excessively onerous either. The most pressing weaknesses lie in mobility and autonomous capabilities. Flaw detection was overzealous at the SVE, with results rife in false positives. Additionally, bugs in the localization module so drastically impacted mobility that we were forced to disable it and rely on dead reckoning. Both of these are issues that can be resolved with more work and calibration. The path planning GUI manages to perform as advertised and is both simple and intuitive to use after minimal instruction.

The chassis and attachment mechanism are both performing well but areas for improvement remain. The center foot broke twice during two tests because of a frail shaft connecting it to the swiveling center segment. Additionally, all of the magnetic feet had a tendency to "pop" off the surface, owing to the fact that magnetic field strength decreases proportionately to the inverse cube of the distance. Consequently, immense forces are necessary to detach a magnet initially, and once detached the necessary force reduces quickly. Since our system using servo motors, the motor will torque as hard as it can to move to the specified location, so one way to mitigate this problem may be to switch to brushless DC motors whose torque can be directly specified. However, it's important to note that the necessary magnetic force, and thus necessary detachment force, is greatly exaggerated in our system which is built for Earth-gravity testing.

The center foot has the least stable attachment of all the segments, with a tendency to slip during turns. Two factors contribute to this, the first of which is that a very precise calibration is necessary to ensure that it remains level at all times and maintains maximum contact with the walking surface. Any deviation and magnetic hold drops off rapidly. Secondly, it has a tendency to slip during turn operations. This is in part due to the first issue causing decreased magnetic force, but also due to the passive degree of freedom that allows it to rotate independently of the center pivot joint. In practice, once magnetic hold is compromised, it has a tendency to rotate about the point of best contact and "roll" down the wall. One possible improvement would be to change the rotation of the foot into an active degree of freedom and control its orientation during robot turns.

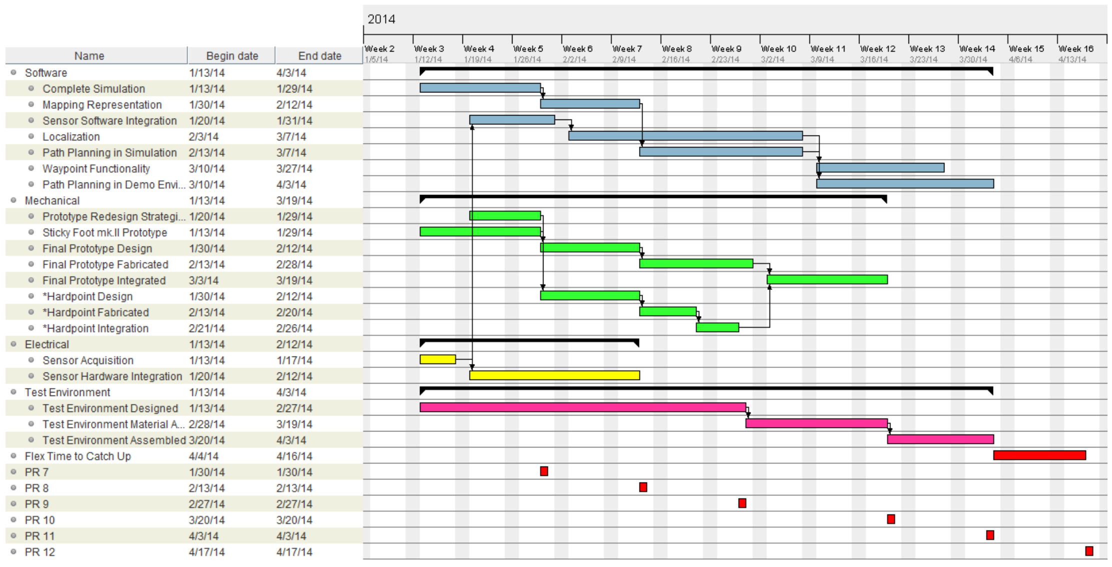
**Table 2: Subsystem strengths and weaknesses**

| | Strengths | Weaknesses |
|---|---|---|
| Power | Able to Power all Motors | Battery duration only 40 minutes |
| Computing | Simple ROS Communication | System boot up lasts an entire minute |
| Chassis/ Mobility | Lightweight, Rapid Fabrication | Chassis weakest in the center joint |
| Attachment | Strong resistance to normal forces | Center segment foot tends to so slip |
| Communication | Simple ROS Communication, wireless radio | Occasional lost packets<br>Occasional connection loss |
| G.U.I | Reliable, simple with mouse-only interaction | Flaw detection is overzealous, often detects shadows as defects |
| Autonomy | Path planner simple and efficient | Localization prohibitively buggy |
| Simulation | Clean kinematic representation | No physical dynamics |

## 8. Project management

### 8.1. Schedule

Our initial spring development schedule with Progress Review milestones is depicted in Figure 29. The primary goals for the spring were the integration of new sensors, including cameras and an IMU, the redesign of the chassis and attachment mechanism, implementation of software localization and path-planning, and the design and construction of a demo test environment. These tasks were roughly partitioned between Software, Mechanical, and Electrical engineering tasks.



| 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 29. Original Development Timeline for the Spring Semester**

In practice, we gave little consideration to scheduling more than two or three weeks into the future, tending instead to reprioritize work in response to test results and implementation hurdles. Scheduling was not maintained during our regular meetings in any official capacity, rather we tended to assign each other immediately necessary tasks to move along the project, trying to do as much work in parallel to keep everyone busy.

Multiple software functionalities were developed in the spring semester, including path planning, localization, and flaw detection. While our projected schedule had planned for these to be carried out one after another, instead they were developed in parallel by three people and only integrated in the final weeks of the semester. In hindsight this was bound to be problematic, owing not only to the typical pitfalls and unexpected issues to be found in integration, but also because the person performing software integration had to read, interpret, and understand a large body of code he'd not written even while that code was still being finished. This was in contrast to the fall semester, when all software had been written and integrated by one person.

The test environment was planned to be designed, built, and assembled over the course of the semester, but in practice was typically improvised shortly before it was needed for testing. Our earlier plans had been for a relatively elaborate rig consisting of a 3D structure to be suspended in the air for the robot to climb around. As the scope of the project was reduced to vertical surface navigation and we switched to a magnetic attachment mechanism, the requirements for the test environment became simply a flat, smooth, ferromagnetic surface.

The biggest time sink by far was the effort directed toward a dry adhesive attachment mechanism. Despite being among the most crucial subsystems of the entire robot, it was treated as a secondary priority until the spring semester, and ultimately proved infeasible for our purposes. Perhaps owing to the urgency of our test schedule, work progressed more quickly after the switch and we produced early prototypes of the magnetic attachment mechanism within a couple weeks.

## 8.2. Budget

In the end, we finished significantly under our projected budget for the project. Our largest expenditure was the purchase of our servos, which came to $875.33 over the two semesters (counting replacements, mounting hardware and spares). But in total we spent only 56% of our total allowable budget.

### 8.2.1 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** | **$1,752.79** | **$447.21** |

### 8.2.2 Spring Expenditures

| Description | Prediction | Total | Difference |
|---|---|---|---|
| Cameras | $150 | $172.48 | -$22.48 |
| IMU | $150 | $17.97 | $132.03 |
| Spring demo environment | $100 | $39.00 | $61.00 |
| Misc. Hardware | $600 | $239.70 | $360.30 |
| **Spring Total** | **$1000** | **$469.15** | **$530.83** |

### 8.2.3 Total Budget

| | |
|---|---|
| Fall Expenditures | $1,752.79 |
| Spring Expenditures | $469.15 |
| **Total** | **$2,221.94** |

## 8.3. Risk management

Risks that were recognized early on are summarized in table 3. We were successful in mitigating many of the risks related to manufacturing and human resources, and less so in the areas of development and planning. The possibility of the adhesive foot failing was underestimated early on, and this oversight resulted in much lost time. Had that risk been recognized as more dire, we may have been more diligent in establishing the adhesive foot's feasibility early and moved on to magnetic feet far sooner. This oversight also interacted with planning issues. Due to the short-term nature of our planning efforts, the prospect of a multi-month delay in attachment mechanism development hadn't occurred to us. In short, we were generally self-assured that the issue would always be solved "in the next development sprint" which was within the next week or two. It wasn't until the spring semester that we realized just how many springs had passed and the solution had failed to materialize. We also failed to carry out our mitigation plan to frequently review and revise our schedule instead opting to focus on “next steps” and gradually lose sight of the overall schedule.

One mitigation plan that proved crucial was our decision at the start of the spring semester to add a generous buffer in our development schedule prior to the SVE. As it turned out, development lagged expectations significantly in multiple subsystems and we used the entirety of that buffer to catch up. Had we not planned this way, we might have filled that time with a larger scope which would ultimately have to be abandoned. Additionally, despite suffering numerous failed components (in particular motors), we were indeed well-stocked with spares of all critical hardware and were never without a drop-in replacement. We were able to achieve this while staying well under budget, and in fact procured almost exactly as many spares which ended up being necessary.

**Table 3: Risk assessment and mitigation plan**

| ID | Risk Description | Category | Root Cause | Mitigation |
|---|---|---|---|---|
| 1 | Adhesive foot mechanism could not support the robot | Development | Material failure in dry-adhesive material<br>Team inexperience | Used magnetic attachment as a backup plan, which was ultimately pursued |
| 2 | Project Damage due to Laboratory Flooding | Development | Spring thaw and rain | Store all sensitive components above desk-level |
| 3 | Electrical failure or burnt components during testing | Electrical | Large power loads from servo motors could overload driver electronics | Spares of critical components and power protection circuits |
| 4 | Team Member Illness | Human Resources | Hygiene, nutrition, chance | Wash hands, plenty of rest (at least one day a week), cross-competency |
| 5 | 3D printer breakdown | Manufacturing | Overuse of 3D printers, poor maintenance, schedule difficulties | Two printers in lab, interchangeable designs, MDF still available as a fallback |
| 6 | 3D printer tolerance/warping issues | Manufacturing | ABS 3d printers are susceptible to bed separation, warping problems if used improperly | 3D printing specialist on the team. Design with 3D printing in mind. |
| 7 | Chassis Flex problems | Mechanical | Strain and flex in linkage design makes it impossible for us to reach our accuracy requirements | Allotted extra time for design, trying different mechanism and fabrication approaches. |

|  |  |  |  |  |
|---|---|---|---|---|
| 8 | Oversights in project pipeline or interdependencies | Planning | Excessive project complexity and scope leads to blind spots in development plan. | Regular review of schedule and overarching development approach |
| 9 | Development Schedule Delays | Planning | Human time limitations and unforeseen technical difficulties | Front-loaded schedule, 4 week flex time before final demos |
| 10 | Schedule Delay due to Laboratory Flooding | Planning | Spring thaw and rain | Locate additional backup workspace above basement-level |
| 11 | Regressions in third-party libraries | Software | Software development outside of our group in sourced third-party libraries causes new bugs or loss of functionality | Version controlled copies of libraries so we can revert as necessary |
| 12 | Failure to integrate third-party libraries into ROS nodes | Software | Complexity of third-party libraries | Include parent projects in team repository. Review package updates and test before committing changes. |
| 13 | Robot falls during wall tests | Testing | Gravity | Human spotter on duty during all vertical tests |

# 9. Conclusions

## 9.1. Lessons learned

### 9.1.1 Project Management

Team meetings were especially effective for keeping everyone up to date on the project status, assigning new tasks, and plotting our path forward. Clearly defined, incremental tasks were much more useful toward fully utilizing our manpower as well as parallelizing design and development. Finally, the use of a centralized repository for documents, code, media, etc. was essential to keeping our body of work well-organized.

Two areas of project management stuck out as problematic. The first was our tendency to over-promise for project goals and milestones. While this may have reflected healthy ambition on our part, it had cascading effects on our prioritization and execution of development. More realistic estimates and a significant time and cost buffer to our estimates may remedy this tendency, as well as soliciting more feedback from experienced mentors in determining these estimates. Another problem was the failure to prioritize the attachment mechanism before all else, leading to its lagged development, despite it being utterly essential to the entire robot's operation. In the future, this may be remedied by increased focus on functional requirements and their interdependencies when planning development.

### 9.1.2 Design Phase

During the design phase, we used a spiral model which allowed for rapid development and frequent reappraisal of the direction of the project. In conjunction with this, using version-controlled design materials allowed us to revert to older designs when necessary as well as keep a record of the design process.

One misstep made early on was designing critical systems around experimental technology. The failed adhesive foot design was a large waste of time and effort, and caused lengthy project delays. Rather we should have started with the simplest commercial solution first and innovated as necessary to solve problems as they arose. If experimental tech is core to the success of the project, we ought to maintain good relationships with the experimenters and share knowledge both ways. Another misstep was our early trade study into similar solutions before we had performed a detailed requirements analysis, leading to a fixation on an existing solution that we later realized was greatly overcomplicated.

### 9.1.3 Procurement

Procurement was generally well-handled throughout the year. Assigning one person to procurement but keeping transparent records, ordering from well-known and reliable vendors, and keeping a stock of spare parts for critical components all served us well. The only hiccups occurred in occasionally unknown shipping statuses and lost packages after delivery. This may be remedied by adding tracking/order numbers to the ordering spreadsheet to help locate these packages.

### 9.1.4 Development

Our most valuable asset during development was by far parallelization. For the most part, we were able to accomplish a great deal of work independently, although this did lead to integration issues due to unfamiliarity with each other's work. One particularly beneficial instance of parallelization was the design/fabrication/assembly/test iteration cycle that we practiced with each of the various robot segments.

However, two recurring issues were communication gaps and unnecessary progress stalls for individuals. These often happened because one person relied on the work of another to continue, or wasn't sure how to progress although another teammate with experience in that realm would easily offer advice. This might be best remedied by regular group work sessions, wherein communication is easy and accountability issues can be resolved quickly.

### 9.1.5 Electrical

Electrical subsystem development benefited greatly from detailed trade studies and design validations being performed prior to ordering and assembling. Issues in electrical systems included sensor noise, which may be remedied with filtering capacitors. Additionally, we over-designed our first iteration of the power distribution, leading to a cumbersome board which failed mass and volume requirements. While this was due in part to functional requirements that were later de-scoped, a large part of it was due to insufficiently detailed requirement analysis leading to a tendency to “play it safe” by building a board with capability to spare.

### 9.1.6 Software

In the realm of software, we found that designing a GUI early on actually doubled as a means of charting our software’s functional requirements and clearly laying out the desired end result. Afterwards, it was a matter of writing the code to wire all the buttons and widgets that had been laid out.  Version control benefited us greatly, as did code reviews when they occurred.

A common issue was code being altered on the eve of a demo, often for non-critical issues, and breaking the overall functionality as a side effect. Maintaining “release” and “development” code branches in our repository would provide a backup in situations like this. Additionally, a problem in late spring was the parallel development of multiple modules that had to be integrated later on by someone with only partial familiarity with each module. More frequent code review, or continuously integrating code even before it’s functionally compete may prevent such situations in the future.

### 9.1.7 Mechanical

We benefited heavily from having all team members trained and able to access machine shops on campus, making task assignment in this area far more flexible than others. Additionally, access to laser cutters and 3D printing made rapid prototyping fast and easy.

On some occasions, however, we discovered defects or instances of flex only after assembling an entire prototype. A good fix for this would be to select materials more carefully early on, and perform incremental design validation tests for subcomponents before assembling entire prototypes.

## 9.2 Starting From Scratch

If we were to start the same project again from scratch, many of the lessons learned both in process and implementation would factor in and a much more robust robot and expanded scope would result. Significant time was spent exploring the overly-complex hexapod design, as well as the dry adhesive option for an attachment mechanism which failed to pan out. Focusing on the inchworm design and a magnetic attachment mechanism from the start may well have led to the same quality of mobility that we have now at the end of the fall semester. With much more time available to analyze and adjust the mobility system, achieving successful plane transitions, mobility on an inverted surface, and perhaps even integration of a rotary tool may have been plausible.

In terms of process, we only began holding meetings twice weekly during the spring semester, and these were immediately beneficial to making continuous progress. While at the end of the project we were still refining our meeting processes and maximizing their returns, we certainly would have benefited from them in the fall semester. A more structured approach to meetings and work sessions, such as scrum, would also be very helpful at keeping the project on track, and planning into the future better.

## 9.3 Future work

The use of magnets in the current prototype was motivated by expediency, in that it would permit us to test and further develop the mobility system without incurring the high costs and lead times associated with high-quality dry adhesive materials. However, very few actual spacecraft hull materials are ferromagnetic. Thus, future investigations into alternative attachment techniques, such as dry adhesives or electrostatic grippers will be necessary to make this robot practical for actual spacecraft maintenance.

Further, to fully realize the potential of this unique mobility system, mapping and path planning must be expanded to cover 3D objects and not simply flat planes. Doing so will allow the robot to perform plane transitions and freely traverse objects much more like a spacecraft. Variants of the attachment mechanism may even enable the traversal of curved hulls. Further, to traverse an actual spacecraft, the robot's environmental awareness must be supplemented with obstacle detection and avoidance so as to not damage or be deterred by equipment or infrastructure protruding from the spacecraft hull.

Localization is currently dependent on the placement of fiducial markers which may not be available on a spacecraft surface, and so a more robust and flexible means of localization will be necessary for real world application. Computer vision, star tracking, pre-made maps, or human guidance may all be useful in establishing reliable localization, and heavy machine learning is certainly feasible given that most computation-heavy tasks can be performed at the robot command center, which may even be located in ground station. Machine learning should also be useful in improving the flaw detection algorithm, which is currently rudimentary and prone to false positives. On the robot's end, more accurate and precise joint position tracking is essential, not only for maintaining a precise mapping of camera views onto the world map, but also for the manipulation of a rotary tool or other equipment which the robot may operate in addition to inspection tasks.

## 10 References

1. Kauderer, A. "“NASA- Canadarm2 and the Mobile Servicing System." Internet: http://www.nasa.gov/mission_pages/station/structure/elements/mss.html(2008).
2. Wagner, Rick, and Hobson Lane. "Lessons learned on the AWIMR project. “Proceedings of the IEEE International Conference Robotics and Automation, Space Robotics Workshop. 2007.
3. Murphy, Michael P., and Metin Sitti. "Waalbot: An agile small-scale wall-climbing robot utilizing dry elastomer adhesives Mechatronics, IEEE/ASME Transactions on 12.3 (2007): 330-338.
4. Kennedy, Brett, et al. "Lemur IIb: a robotic system for steep terrain access. “Industrial Robot: An International Journal 33.4 (2006): 265-269.
5. Dobrowolny, M., and N. H. Stone. "A technical overview of TSS-1: the first tethered-satellite system mission." Il Nuovo Cimento C 17.1 (1994): 1-12.
6. http://github.com/ros-drivers/rosserial
7. http://github.com/SpaceJockey/ipcam_ros
8. http://wiki.ros.org/ar_track_alvar
9. http://wiki.ros.org/cv_bridge