Carnegie Mellon University

Robotics Capstone Project

System Readiness Review

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. Cameron Riviere

Version 1.0
April 5, 2017

Contents

List of Figures

1 Build Progress

This section details system development and progress made since the last milestone presentation. Progress is split into two major sections: electromechanical and software. Electromechanical updates detail chassis build progress, as well as setup of the electronics to drive the motors for locomotion and using the writing implement. Software updates describe progress towards subsystem completion.

1.1 Electromechanical Updates

As shown below in Fig.1, we have built a physical robot prototype that incorporates chassis, painting mechanism, and locomotion system. The chassis is made of laser patterned acrylic. It is designed to be compact, as possible because smaller robots are less likely to collide with each other during drawing operations. This prototype proves that the chassis' current cutout sizes have no clearance issues with moving components.

CAD/full_system_1.jpeg

CAD/full_system_2.jpeg

Figure 1: Prototype Overview, from left view (left) and right view (right)

The painting mechanism is composed of a 3D printed chalk holder and a micro gear motor which is shown in Fig.2. The driving motor is mounted to the chassis via an off-the-shelf motor case. When designing the chalk holder, four internal ribs were added inside the holder to securely hold the chalk marker in place. This also allows users to easily switch out the marker. A thin cap is added on the bottom of the chalk holder to prevent the chalk marker from sliding out while drawing. One flaw of this design is that the holder's D-shaft cutout is slightly undersized. As a result, the chalk holder broke while pressing the motor shaft through the holder. This problem will be addressed in the next iteration.

Fig.3 shows the locomotion system. Four Mecanum wheels are oriented in a "X" shape to minimize motor workload. These wheels are connected to driving motors through 3D printed wheel adaptors. These adaptors contain two segments: a standard Lego technic axle and D-shaft housing. Like the chalk holder, the D-shaft cutout is a little undersized. Therefore, we had to press fit the motors in.

Besides mechanical update, motor controller code was also completed. However, we did not get enough time to wire all electronics to this prototype and test the code. This would be the next step of system development.

Since we have enough left-over budget, we plan to use 80/20 aluminum frames, instead of wood, to construct the camera jig. The jig will be built using components listed in Fig.4. We are in the process of testing camera's optimal height, and will then incorporate that information to the camera jig CAD design.

1.2 Software Update

We detail the software progress made across the following subsystems. Many subsystems are near full development, allowing us to begin integration.

CAD/painting.jpeg

CAD/chalk_holder.jpeg

Figure 2: Painting Mechanism (left), Chalk Holder CAD (right)

CAD/wheels_1.jpeg

CAD/wheels_2.jpeg

CAD/wheel_adapter_1.jpeg

Figure 3: Locomotion System (left), Locomotion System Components (center), Wheel Adaptor CAD (right)

1.2.1 Communication

The goal of the communication subsystem is to abstract out networking operations such that other subsystems can maintain modularity. Currently, this subsystem is mostly complete. The system is able to easily establish, maintain, and close TCP connections between the separate robots, given their IP addresses. It also contains code to generate a singular protobuf data containing all data necessary to send robots, and pass it through the TCP connection. Onboard communication code is able to continually receive these TCP protobuf messages, and parse them accordingly.

In order to further isolate communication code from the actual subsystems, other subsystems fill in a data struct containing relevant information to send to the robots. For example, the localization subsystem will enter information into a localization data struct, which is passed to the communication subsystem at runtime. The communication subsystem will then parse relevant localization data into the protbuf message to send across the network. These data structs have the additional use of allowing for convenient transfer of data between other subsystems as well.

1.2.2 Locomotion

The locomotion subsystem has had some major changes. Previously, we planned to run locomotion offboard, where it would generate motor powers to send to the robot system. However, analysis showed that offboard motor processing incurs a higher latency than an accurate control could easily use. Decreased latency allows the motor PID controller to provide more accurate stabilization, to better enable the robot

CAD/8020.jpeg

Figure 4: 80/20 Parts

to follow a path. As a result, the locomotion subsystem is being moved to an onboard robot system. The offboard system will send the robot's current position and orientation, as well as a target position and orientation. The robot will compute the locomotion commands necessary to reach the target, and run the position and velocity controller accordingly. The positional localization and target data is able to be sent via protobuf message to the onboard system. The encoder and localization motor control is written, and will be tested with sample data once chassis construction is completed.

1.2.3 Localization

The localization subsystem is mostly unchanged, and currently in progress. Integration of the AprilTags C++ library is in progress, which requires setting up the C++ environment, and passing functions to the Python subsystem via Boost.

1.2.4 Scheduling, Distribution and Planning (SDP)

In order to our SDP module, we first had to add a few basic UI elements. We laid out a file format for specifying the lines to be drawn and wrote the UI functionality to parse the data in.

Given the data the next step is to distribute the work between the two robots, offline. We will later describe a first pass distribution algorithm along with the UI developed to visualize its results. We will conduct further testing to see if a more advanced algorithm is needed. Luckily, this can be done in parallel with other developments since we have fixed the input and output of the system, allowing us to swap in different distribution tactics. The output of the distributor is a set of vectors that specify the plan for each robot. These vectors will then be handed off to the locomotion module, described above, that will follow each of them in sequence. Therefore, this gives us two next steps: to integrate the planning with the locomotion and to develop a collision avoidance strategy.

To handle collisions, we will start off with a naive strategy. We define a robot's *boundary* as a fixed radius circle around robot, where the radius exceeds that of the robot to provide cushion. As each robot moves, it will check if the other robot's boundary intersects with its own boundary. If this condition is true, one robot (Bad) will stop execution, allowing the other robot (Blue) to pass until the condition is false. While we believe this method will always prevent collisions, it may not be the most efficient. Therefore we will implement this and test accordingly to check performance.

For our distributor, we developed a very greedy method. We start Blue the robot at one corner of the drawing area and Bad the robot at the other corner. Our goal is greedily balance their cost, where cost corresponds to the length of the line drawn so far. We initialize both robots with cost zero. From there, we loop over the line count. We pick the robot with the lower cost, defaulting to one in the case of equality. Whichever robot has the lower cost, we pick the line with the closest starting point to the robot's current position. We then calculate the cost as the distance to drive to the line plus the distance to drive to draw out that line. Having updated the robot's position, we continue.

To illustrate the output of our planner we developed a visualization, shown below in Fig.5. The red path represents Bad the robot and the blue path represents Blue the robot. Solid lines corresponding to drawing lines and hence making a mark on the pavement while dotted lines correspond to purely transit. In Fig.5, Blue the robot starts from the top right corner and transits to draw one line and then return home. In contrast, Bad the robot draws two intersecting, nearby lines.

figs/figure_1.png

Figure 5: Planning Output

In some cases, such as in Fig.6a, we see a nice breakdown of work as each robot is responsible for one box. The occurs because at the end of each step, the next closest line is directly adjacent, making the transit distance zero. In Fig.6b we see that the greedy approach segments the lines less cleanly and raises the possibility of collision. We note that at every intersection of two robots there is a possibility of collision, but not necessarily if the robots traverse those areas at different times. Additionally the robots could collide outside of intersections since we are not dealing with point robots. As mentioned above, we will continue to explore collision avoidance.

figs/figure_7.png

(a) Good Distribution

figs/figure_4.png

(b) Plan with Possible Collisions

Figure 6: Example Planner Output on Test Cases

2 Project Management

2.1 Work Breakdown Schedule

In this section, we present the Work Breakdown Schedule for the project.

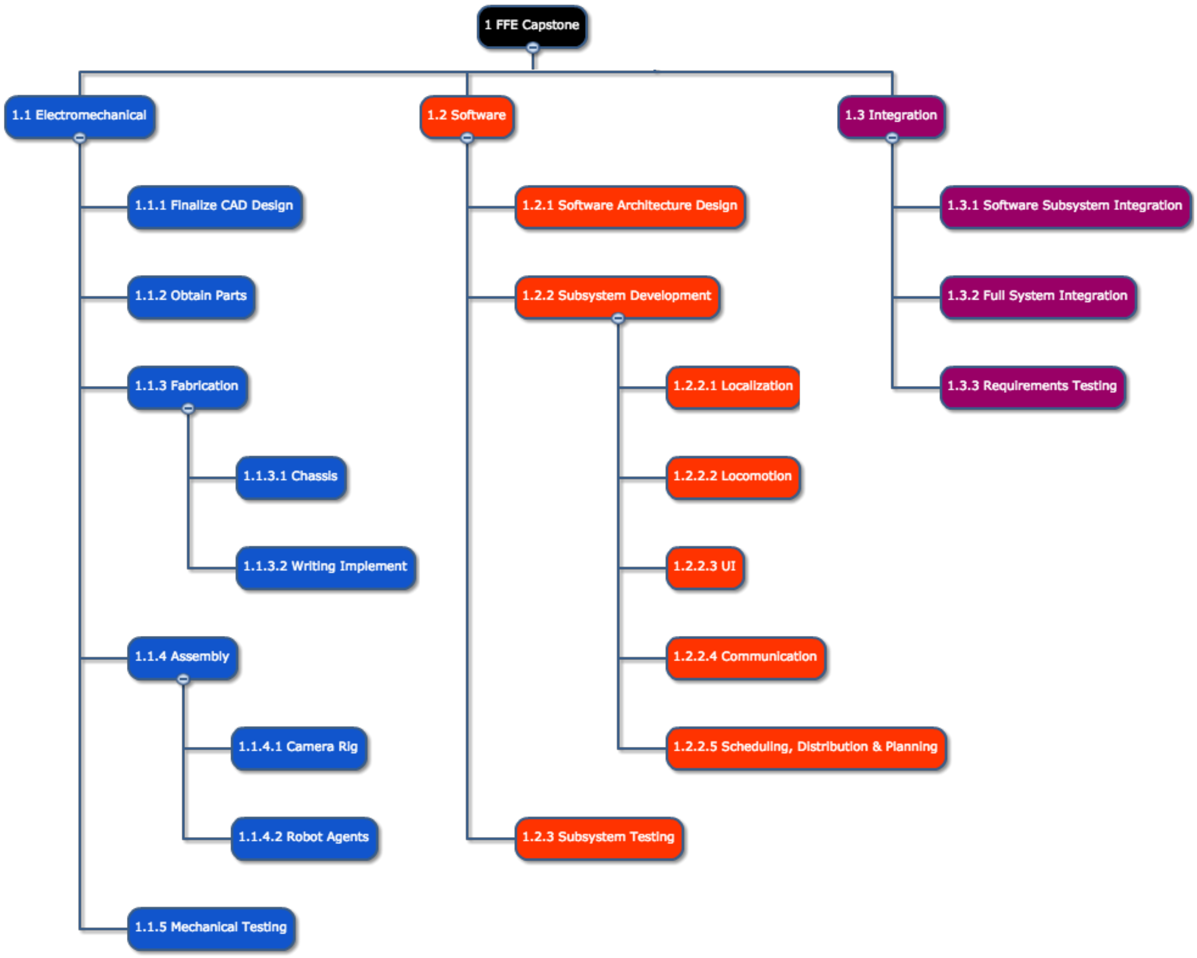

Figure 7: Full WBS for the project

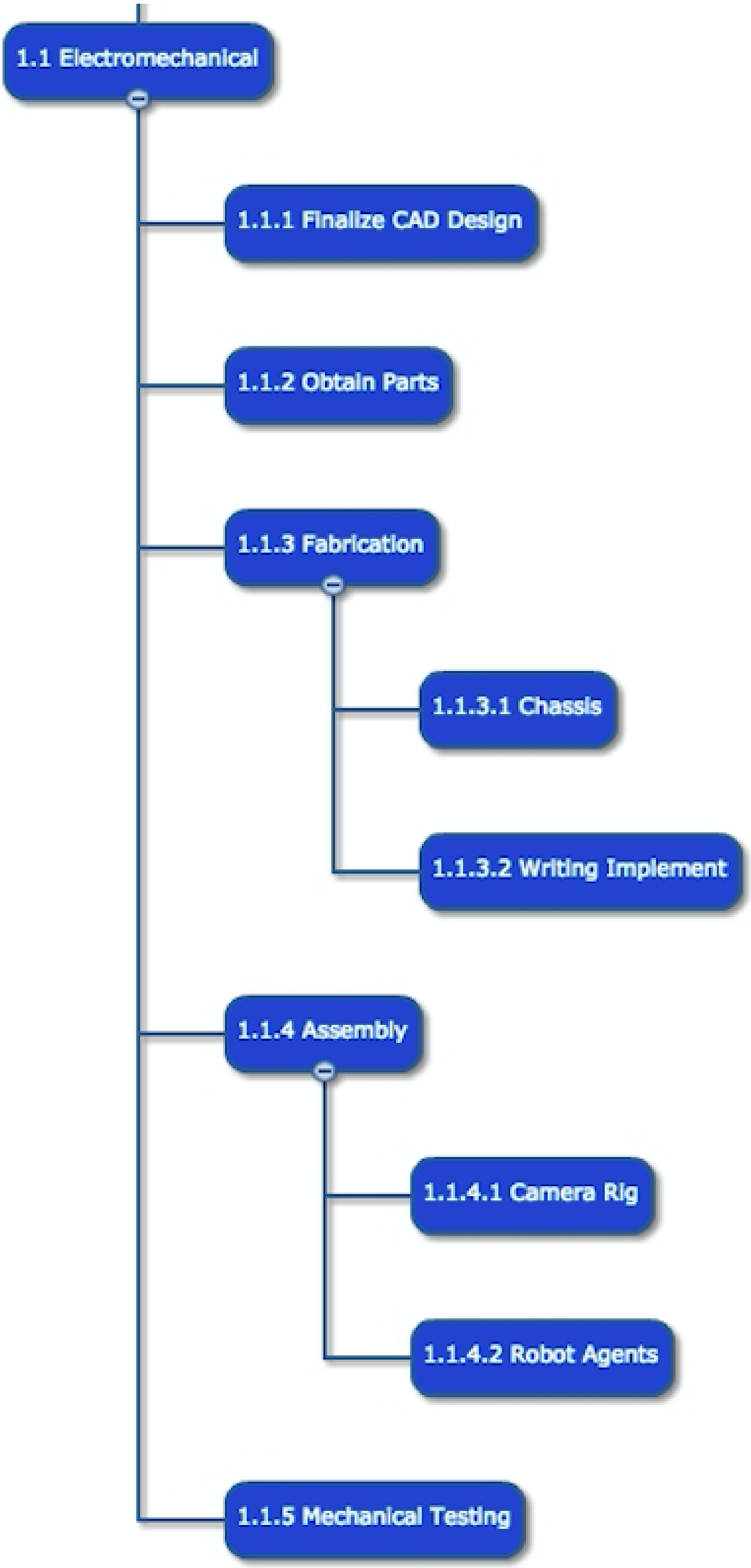

Figure 8: Electromechanical WBS section

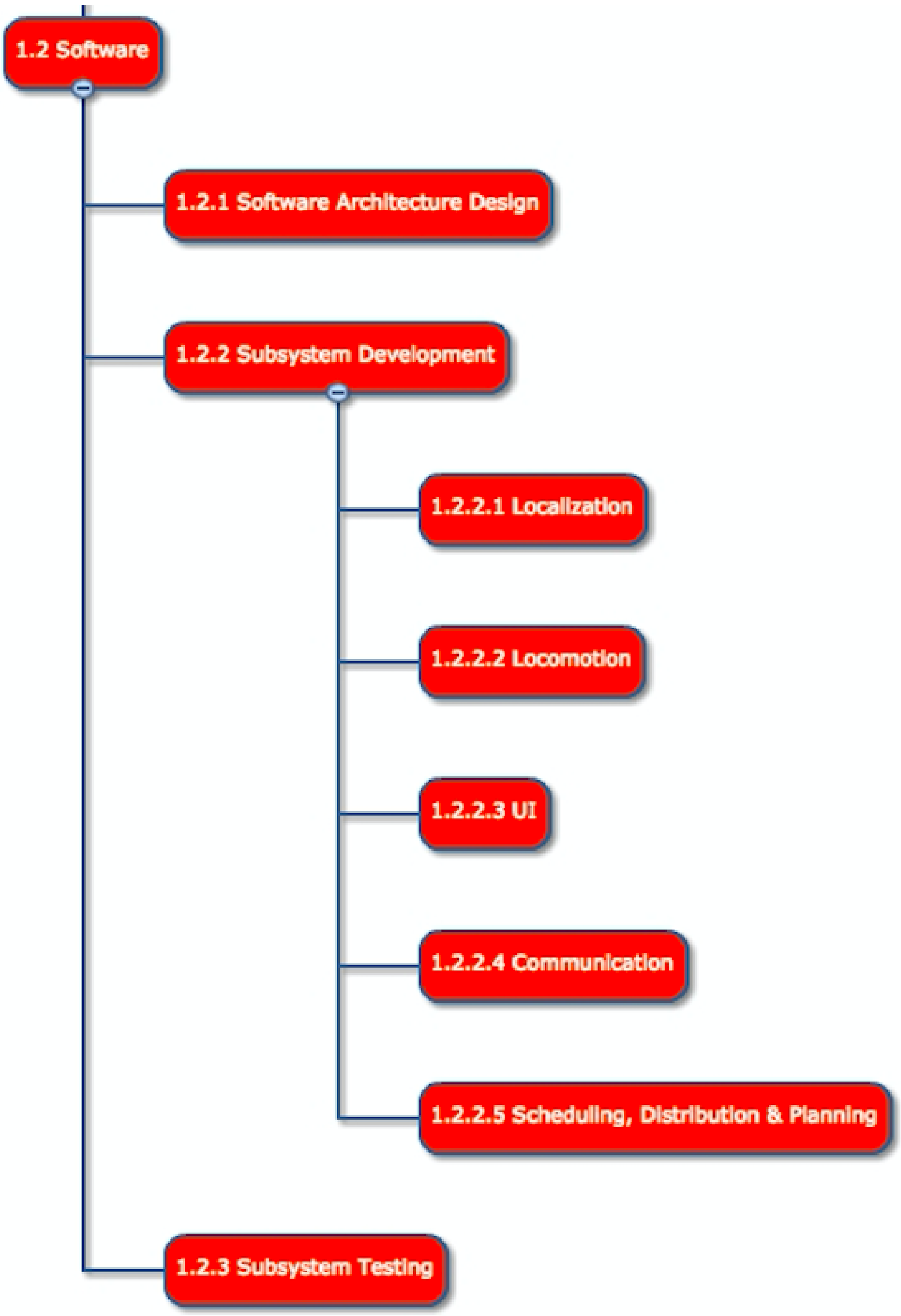

Figure 9: Software WBS section

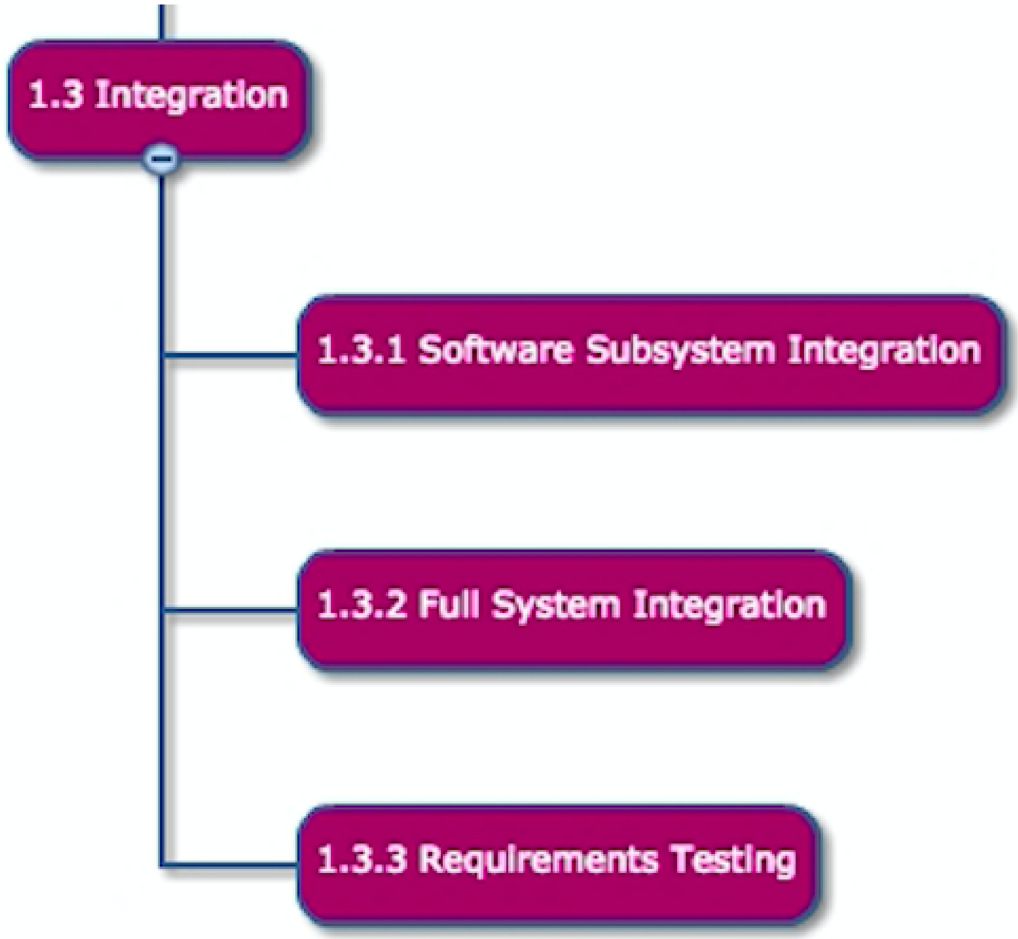

Figure 10: Integration WBS section

The WBS dictionary entries include more information on each of the work elements of the project. Information such as estimates for the amount of time each task will take and their dependencies will help us adhere to our schedule, while determining the owner of each task will improve tractability of the workflow.

WBS#:	1.1.1	Task:	Finalize CAD Design
Est. Effort (hrs):	3	Owner:	Eric
Resources:	CAD software	Work products:	CAD files
Description:	Update the CAD design to fit the final design of the robot		
Input:	Previous designs, new design ideas and requirements		
Dependencies:	Complete design review		
Risks:	Designs cannot be completed on time		

WBS#:	1.1.2	Task:	Obtain Parts
Est. Effort (hrs):	20	Owner:	Don
Resources:	Parts list	Work products:	Parts order receipt
Description:	Finalize the parts list, and contact the necessary people to ensure that parts are ordered		
Input:	CAD designs, electronics designs		
Dependencies:	Finalized CAD designs		
Risks:	Parts ordering procedure is more time consuming than expected		

WBS#:	1.1.3.1	Task:	Fabricate Chassis
Est. Effort (hrs):	4	Owner:	Eric
Resources:	CAD designs, MechE shop	Work products:	Chassis components
Description:	Use the Mechanical Engineering machine shop to fabricate components necessary to build the chassis		
Input:	CAD designs, parts		
Dependencies:	Obtain parts		
Risks:	Machine shop is not available, injury from operating machines		

WBS#:	1.1.3.2	Task:	Fabricate Writing Tool
Est. Effort (hrs):	4	Owner:	Eric
Resources:	CAD designs, MechE shop	Work products:	Writing tool components
Description:	Use the Mechanical Engineering machine shop to fabricate components necessary to build the writing implement		
Input:	CAD designs, parts		
Dependencies:	Obtain parts		
Risks:	Machine shop is not available, injury from operating machines		

WBS#:	1.1.4.1	Task:	Assemble Camera Rig
Est. Effort (hrs):	3	Owner:	Don
Resources:	Scrap wood	Work products:	Camera rig
Description:	Build the rig used to hold the camera for the vision system above the drawing space		
Input:	Measurements from demo space		
Dependencies:	Confirmation of demo space location		
Risks:	No extra wood is available, demo space does not have adequate room for the camera rig		

WBS#:	1.1.4.2	Task:	Assemble Robot Agents
Est. Effort (hrs):	5	Owner:	Eric
Resources:	Tools, fasteners	Work products:	Two robot agents
Description:	Use fabricated components to build the two robot agents in the system		
Input:	Fabricated components		
Dependencies:	Fabricate chassis and fabricate writing tool		
Risks:	Parts are broken during assembly, extra parts or fasteners are needed		

WBS#:	1.1.5	Task:	Mechanical Testing
Est. Effort (hrs):	3	Owner:	All
Resources:	Tools, fasteners	Work products:	Two robot agents
Description:	Perform mechanical testing on the robots in accordance with our testing guidelines		
Input:	Mechanically complete robots		
Dependencies:	Assemble robot agents		
Risks:	Tests are failed, and significant time or extra resources are needed to correct the tests		

WBS#:	1.2.1	Task:	Software Arch. Design
Est. Effort (hrs):	3	Owner:	All
Resources:	None	Work products:	Function headers
Description:	Design function I/O, and create function headers for all files we will use in the robot		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Complete design review		
Risks:	Selected software libraries have compatability issues		

WBS#:	1.2.2.1	Task:	Localization Subsystem
Est. Effort (hrs):	6	Owner:	Neil
Resources:	AprilTag library	Work products:	Working localization
Description:	Fill in the function headers for the localization system to develop an end-to-end localization solution for the robots		
Input:	Function headers and design for localization system		
Dependencies:	Software architecture design		
Risks:	Localization system or library is unable to perform to expectations		

WBS#:	1.2.1	Task:	Locomotion Subsystem
Est. Effort (hrs):	5	Owner:	Don
Resources:	Adafruit Motor controller library	Work products:	Control system for motors, robust motion model
Description:	Create a complete set of functions that can be used to direct the robots around the workspace		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Interfacing issues with motors, damanged electronics hardware, unreliable motion models		

WBS#:	1.2.1	Task:	User Interface Subsystem
Est. Effort (hrs):	4	Owner:	Rachel
Resources:	Various UI libraries	Work products:	User interface including calls to other subsystems
Description:	Create a visually appealing and intuitive user interface for the robot system		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Libraries are not available		

WBS#:	1.2.2.4	Task:	Comunication
Est. Effort (hrs):	8	Owner:	Neil
Resources:	Wireless comm. libraries	Work products:	Functions for sending info. back and forth from robots
Description:	Create a reliable communication system between the robots and the central data processing unit		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Wireless hardware is unreliable or interfaces poorly with other software or hardware		

WBS#:	1.2.2.5	Task:	SDP Subsystem
Est. Effort (hrs):	15	Owner:	Rachel
Resources:	SDP research, implementations	Work products:	Complete SDP functions
Description:	Create a flexible scheduling, distribution, and planning subsystem that efficiently assigns work to robots		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	SDP algorithms are not efficient enough to meet requirements		

WBS#:	1.2.3	Task:	Subsystem Testing
Est. Effort (hrs):	4	Owner:	All
Resources:	Software subsystems	Work products:	Complete software subsystems
Description:	Test all software subsystems to ensure that they give the expected output when provided with testing inputs		
Input:	Completed software subsystems		
Dependencies:	All software subsystem tasks		
Risks:	Software subsystems were implemented incorrectly and do not perform to expectations		

WBS#:	1.3.1	Task:	Software Integration
Est. Effort (hrs):	3	Owner:	All
Resources:	Software subsystems	Work products:	Complete software pipeline
Description:	Test integration of all software components by creating an end to end pipeline consisting of all software subsystems		
Input:	Completed and individually verified software subsystems		
Dependencies:	Subsystem testing		
Risks:	Subsystems cannot integrate with each other		

WBS#:	1.3.2	Task:	Full System Integration
Est. Effort (hrs):	3	Owner:	All
Resources:	S.W. and H.W. subsystems	Work products:	Working robot system
Description:	Complete integration of software components with hardware components		
Input:	Completed and individually verified software and hardware subsystems		
Dependencies:	Software integration		
Risks:	Software and hardware cannot interface with one another, models do not work in in practice		

WBS#:	1.3.3	Task:	Requirements Testing
Est. Effort (hrs):	5	Owner:	All
Resources:	Working robot	Work products:	Complete, working robot system
Description:	Verify the reliability and effectiveness of the robot by conducting our full testing suite		
Input:	Unverified but working robot system		
Dependencies:	Full system integration		
Risks:	Robot fails tests, need to rework some subsystems		

2.2 Schedule

NJ: Add in notes about schedule

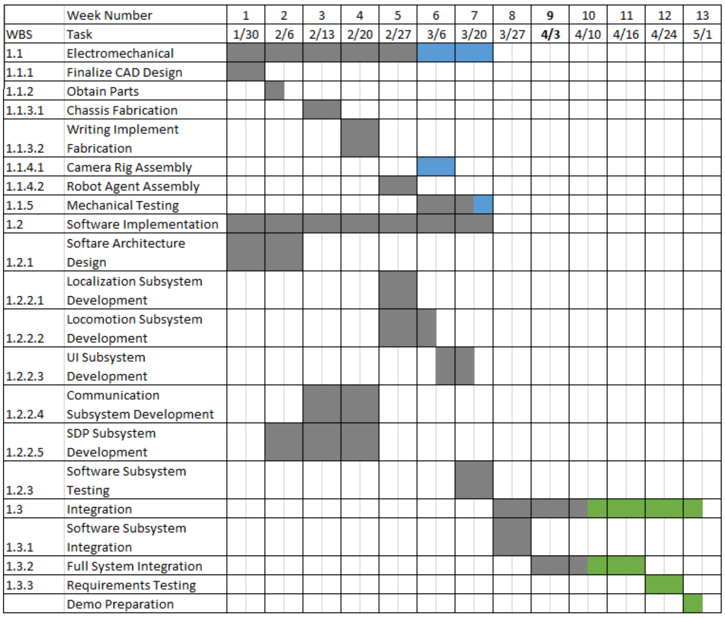

Figure 11: Schedule via Gantt Chart

3 Requirements Tracking

3.1 Objectives Tree

NJ: Insert in images and some comment

3.2 Requirements Traceability Matrix

NJ: Insert images and some comment

4 Risk Management

DZ: Don update this with new information

In this section, we enumerate the risks that our project involves in the form of risk tables. Each table entry details the relevant information about a risk, and explains our plans for mitigating that risk. It also visually displays the risk in the likelihood and consequence diagram.

Risk ID:	Risk Title:	Risk Owner:
1	Defective Parts	Don
Description:		
Parts that we ordered arrived defective or do not perform to specifications		
Consequences:	Risk Type:	
We need to reorder parts, expending extra time and budget	Parts	
Risk Reduction Plan:	Expected Outcome	
We will order only parts that have been extensively reviewed, or we have experience with, and order extra parts	We will be able to properly deal with any parts that break during the development process	

Consequence						
1	2	3	4	5		
					5	
					4	
					3	
	X				2	
					1	Likelihood

Risk ID:	Risk Title:	Risk Owner:
2	Unvavailable Group Member	All
Description:		
A group member becomes unavailable for work due to travel, sickness, or other emergencies		
Consequences:	Risk Type:	
Work that would have been distributed to that group member needs to be reassigned	Logistical	
Risk Reduction Plan:	Expected Outcome	
We will ensure that every group member is always on the same page about progress so we don't lose too much progress	If a member becomes unavailable, it will only be for a short time and can be easily dealt with	

Consequence						
1	2	3	4	5		
					5	
					4	
			X		3	
					2	
					1	Likelihood

Risk ID:	Risk Title:	Risk Owner:
3	Breaking parts	Eric
Description:		
Parts unexpectedly break as a result of accidents or improper use		
Consequences:	Risk Type:	
We need to reorder parts, expending extra time and budget	Parts	
Risk Reduction Plan:	Expected Outcome	
We will practice safe procedures when working with parts and order extras in case	Few parts will break, and even if they do we will have extras on hand	

Consequence						
1	2	3	4	5		
					5	
					4	
					3	
			X		2	
					1	Likelihood

Risk ID:	Risk Title:	Risk Owner:
4	Mecanum Drive Too Unstable	Eric

Description:
The drive mechanism for the robot proves too be too unstable or unreliable for our purposes

Consequences:	Risk Type:
We will need to redesign the drive mechanism, expending considerable time and effort	Design flaw
Risk Reduction Plan:	**Expected Outcome**
We will build enough time in our schedule to deal with it if necessary, and will use suspension	The instability resulting from the wheels will be manageable

Consequence	1	2	3	4	5		
						5	
						4	
				X		3	
						2	
						1	Likelihood

Risk ID:	Risk Title:	Risk Owner:
5	Localization not precise enough	Neil

Description:
Our localization system is not precise enough to ensure that the drawings are accurate representations of input

Consequences:	Risk Type:
We will need to redesign the localization system or redefine drawing requirements	Design flaw
Risk Reduction Plan:	**Expected Outcome**
We will test the localization system early on in order to catch any design flaws within the system	Localization will work well enough for our purposes

Consequence	1	2	3	4	5		
						5	
						4	
						3	
			X			2	
						1	Likelihood

Risk ID:	Risk Title:	Risk Owner:
6	Unexpected Budget Overruns	Rachel

Description:
We unexpectedly run out of budget, because parts cost more than expected or other parties reduce our budget

Consequences:	Risk Type:
We need to scale down our project, or possibly even acquire funds through other means	Logicstical
Risk Reduction Plan:	**Expected Outcome**
We will leave a significant buffer in our budget in case unexpected situations occur	We will have a large enough buffer that essential components will be acquired

Consequence	1	2	3	4	5		
						5	
						4	
						3	
		X				2	
						1	Likelihood

5 Testing and Evaluation Plan

YJ: Eric update this with new information