

Carnegie Mellon University

16-650

System Engineering and Project Management

---

# Conceptual Design Review Report

## COBORG

---

Team C:

Gerald D'Ascoli | Jonathan Lord-Fonda | Yuqing Qin | Husam Wadi  
Feng Xiang

Sponsor:

Biorobotics Lab

December 15, 2020



## Table of Contents

<b>Project Description</b>	<b>1</b>
<b>Use Case</b>	<b>1</b>
Narrative	1
Graphical Representation	2
<b>System-level Requirements</b>	<b>2</b>
Mandatory Requirements	2
Mandatory Performance Requirements	2
Mandatory Non-Functional Requirements	3
Desirable Requirements	3
Desirable Performance Requirements	3
Desirable Non-Functional Requirements	3
<b>Functional Architecture</b>	<b>4</b>
<b>System and Subsystem-Level Trade Studies</b>	<b>5</b>
System-Level	5
Target Detection	6
Hand Detection	7
Voice Recognition	8
Actuated Manipulation - Motion Planning	9
<b>Cyberphysical Architecture</b>	<b>11</b>
<b>Subsystem Descriptions</b>	<b>12</b>
Hardware Framework	12
Electrical Framework	12
Sensors	13
Perception: Vision	13
Perception: Voice	13
Actuated Manipulation	14
<b>Project Management</b>	<b>14</b>
Work Plan and Tasks	14
Key Milestones & Schedule	17
System Validation Experiments	18
Spring Validation Demonstration	18
Fall Validation Demonstration	19
Team Member Responsibilities	20
Parts List and Budget	20
Risk Management	20
<b>References</b>	<b>23</b>
<b>Appendix</b>	<b>25</b>

## 1. Project Description

Automotive and airplane manufacturing can often cause strain to the operator's arms after working for long periods of time, and often require a second person to assist them as they work [1] [2]. Specifically, the attachment of under-wing panels requires two individuals, one to hold the panel, and the other to attach it. A robotic backpack system could hold the panel for its wearer, allowing the operator to use their hands to secure the part. This Collaborative Cyborg Backpack Platform (COBORG) would allow a single individual to accomplish the entire task by themselves, and alleviate the stress and strain induced throughout the workday. To accomplish this operational objective, the COBORG would have to be simple, accurate, and hands-free.

## 2. Use Case

### 2.1. Narrative

Sally works on the assembly of airplane wings at Boeing. When she gets to work she checks her to-do list and sees that her list is topped with a series of tricky assemblies. Knowing that she will require aid shortly, she walks over to the COBORG backpack arm station and signs out one of the units. Picking the unit up from its charging station, she straps the backpack on, adjusting the straps for comfort, and heads over to the plane she will be working on today. After completing some remedial tasks she is ready to move onto the trickier cases where she will require the arm's help. She switches on the backpack arm, which has been in a compact position and not using energy up until this point. Sally grabs the part she requires assistance with and holds it up over her head, fitting it into place (see Figure 1[A]). When the part is stabilized, she says, "COBORG, hold it," and the COBORG backpack arm detects her hands in 3D space and moves the robot arm to a position where it can push on the part and stabilize it (see Figure 1[B]). Sally lowers one of her arms from the part, now that the COBORG backpack arm is holding it, and uses a drill with her free hand to screw the part into place (see Figure 1[C]). While Sally's body shifts its position, the COBORG backpack arm adjusts to maintain the position of the end effector in 3D space, supporting the part regardless of Sally's position within the limits of the arm, with 5 degrees of freedom. Now that the part is fastened, Sally says, "COBORG, return home." The COBORG backpack arm returns to its compact position and goes into sleep mode, awaiting further instruction with minimal power usage. Sally finds the next part for the next task and uses the backpack arm to help her secure it as well. After completing a series of similar tasks, Sally comes to a panel she will have to connect located in a dark area of the interior. Sally quickly switches the end effector of the COBORG backpack arm from the support paddle to the gripper, handily attached to the side of the backpack. She then locks her flashlight into the gripper, moves the arm into place so that she can see the necessary location, and says, "COBORG, stay." The COBORG robot arm maintains its position and orientation in 3D space while Sally moves around, reaching for the panel she can now see, and finishes her task. She then says, "COBORG, return home," removes the flashlight from the gripper, and switches back to the support paddle for her next series of tasks. After completing all of her tricky tasks for the day, Sally returns the COBORG backpack arm to its charging station and signs it back in. While Sally completes the rest of her work for the day, the COBORG backpack arm charges, awaiting its next user.

## 2.2. Graphical Representation

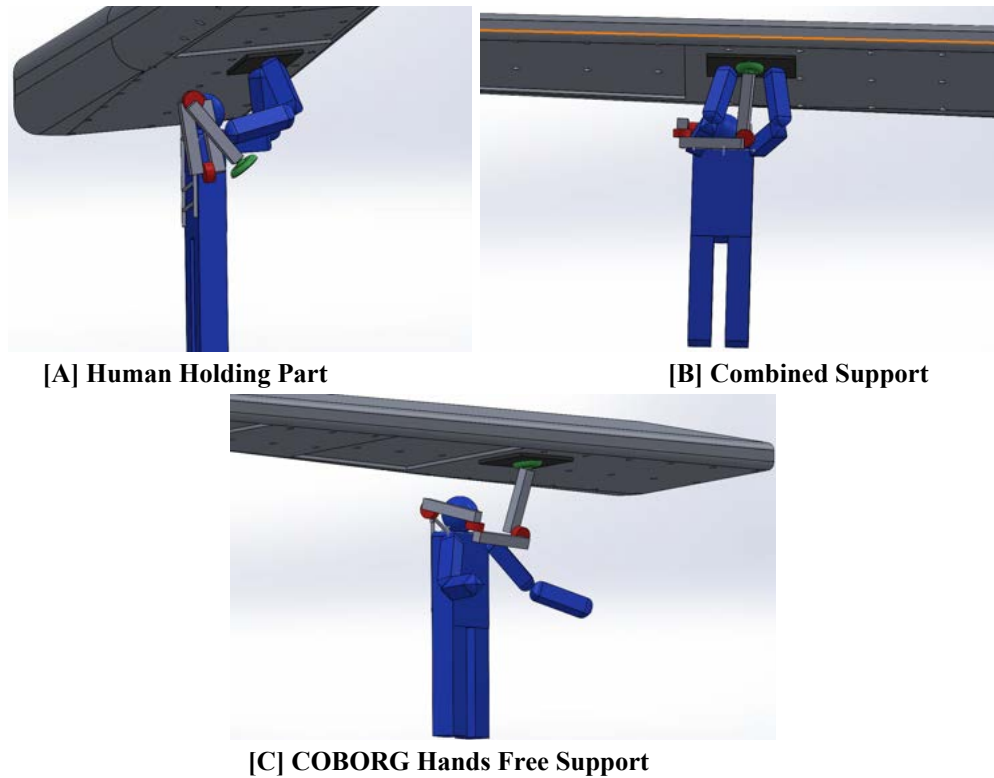


Figure 1 - Graphical Representations of Use Case

## 3. System-level Requirements

### 3.1. Mandatory Requirements

#### 3.1.1. Mandatory Performance Requirements

ID	Requirement
P.M.1.1	(Detect indicated parts) Will have 60% accuracy of detecting indicated part position in 3D space.
P.M.1.2	(Calculate object) Will detect intended object within 5 seconds of when the move command is issued.
P.M.2	(Move to object) Will reach within 12 in. of the planned target position 60% of the time.
P.M.3.1	(Hold object) Will maintain the target's spatial position with less than 12 in of error margin.
P.M.3.2	(Hold object) Will lift at least 2 lbs to full horizontal extension.
P.M.4.1	(Voice command) Will be able to understand the voice command 60% of the time.
P.M.4.2	(Voice command) Will be able to understand at least 2 unique voice commands.
P.M.4.3	(Voice command) Will be able to understand commands of at least 2 words in length.
P.M.5	(Release object) Will release object within 5 seconds of when the release command is issued.
P.M.6	(Compact arms) Will bring the full robot arm to within 20 in. of the point of attachment to the backpack.

Table 1 - Mandatory Performance Requirements

### 3.1.2. Mandatory Non-Functional Requirements

<b>ID</b>	<b>Requirement</b>
N.M.1	Will be ergonomic for spinal comfort.
N.M.2	Will weigh less than 40 lbs.
N.M.3	Will be aesthetically pleasing.
N.M.4	Will operate safely.
N.M.5	Will be simple to operate.
N.M.6	Will be able to perform untethered for 20 minutes.
N.M.7	Will require minimal part modification.
N.M.8	Will be operable on a portable computer.

**Table 2 - Mandatory Non-Functional Requirements**

### 3.2. Desirable Requirements

#### 3.2.1. Desirable Performance Requirements

<b>ID</b>	<b>Requirement</b>
P.D.1.1	(Detect occluded parts) Must be able to detect part while 20% of the part is occluded.
P.D.1.2	(Texture Invariant) Must be invariant to part texture, specifically matte finish and gloss finish.
P.D.1.3	(Pose Detection) Shall detect the orientation of the part (x,y,z,w,p,r) with error no greater than 45°.

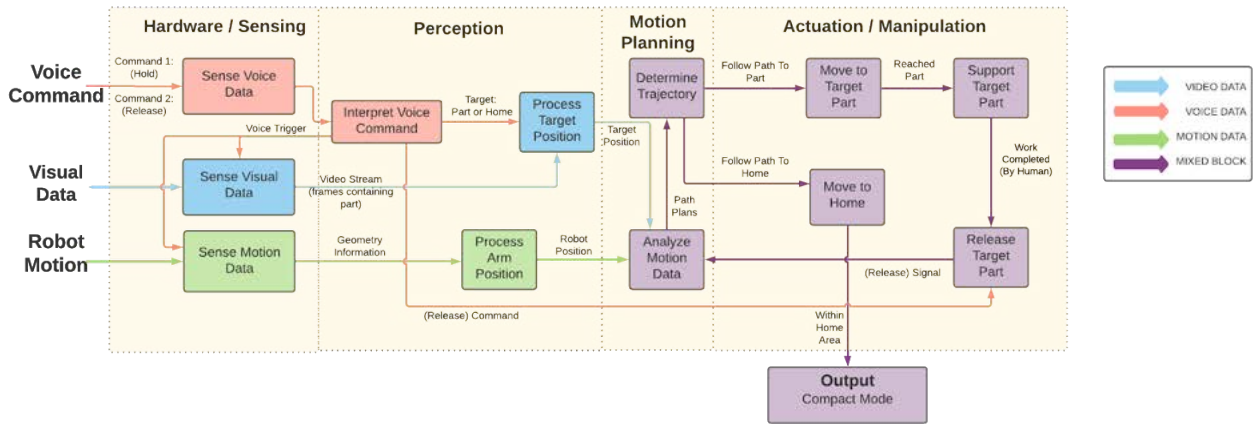
**Table 3 - Desirable Performance Requirements**

#### 3.2.2. Desirable Non-Functional Requirements

<b>ID</b>	<b>Requirement</b>
N.D.1	Will be able to operate standalone (no WiFi).

**Table 4 - Desirable Non-Functional Requirements**

## 4. Functional Architecture



**Figure 2 - Functional Architecture of COBORG System**

The Functional Architecture shown in Figure 2 demonstrates the major functions of our system and the data flow between the subsystems. It contains three aspects: the input data, output mode, and the four major subsystems. The input data all come from the hardware sensors on the COBORG system. Our system will interpret the data and generate a manipulation output through the robot arm. The details for each subsystem are introduced in the following paragraphs.

The System Inputs, as listed on the left side in Figure 2, is the data coming from the hardware (sensors) on the COBORG. Specifically, our system takes three kinds of input data: Voice Command, Visual Data, and Robot Motion. The voice commands from the user is captured by the microphone on the robot backpack. For the visual input, our system perceives the target object from the camera sensor on the COBORG backpack. The motion data is the third type of input that our system will use to analyze the robot arm motion information: position, velocity, acceleration, and other IMU inputs.

The Hardware and Sensing Subsystem is responsible for capturing the inputs. This subsystem will begin operation when voice commands come into the system. Based on the voice command trigger, other sensors will capture visual and motion data. The video stream and the geometry information will be fed into the Perception Subsystem.

The Target Perception Subsystem receives sensing data from the Sensing Subsystem. After the system interprets the voice command from the Sensing Subsystem, based on the voice command content, the system will detect the desired target (part, home) position and retrieve the robot arm motion data to execute Motion Planning.

The Motion Planning Subsystem will use the motion data and the target position information to determine the trajectory for the robot arm. The data will first come into the Analyze Motion Data block to generate possible path plans. The determined trajectory from multiple path plans will be forwarded to the Actuation and Manipulation Subsystem.

The Arm Actuation and Manipulation Subsystem receives the trajectory as the input, and by controlling the robot arm, COBORG will follow the trajectory, move to the desired object, and hold the object overhead. Once the voice command trigger is received, the robot arm shall release the object and move back to the Compact Mode, which is the system output.

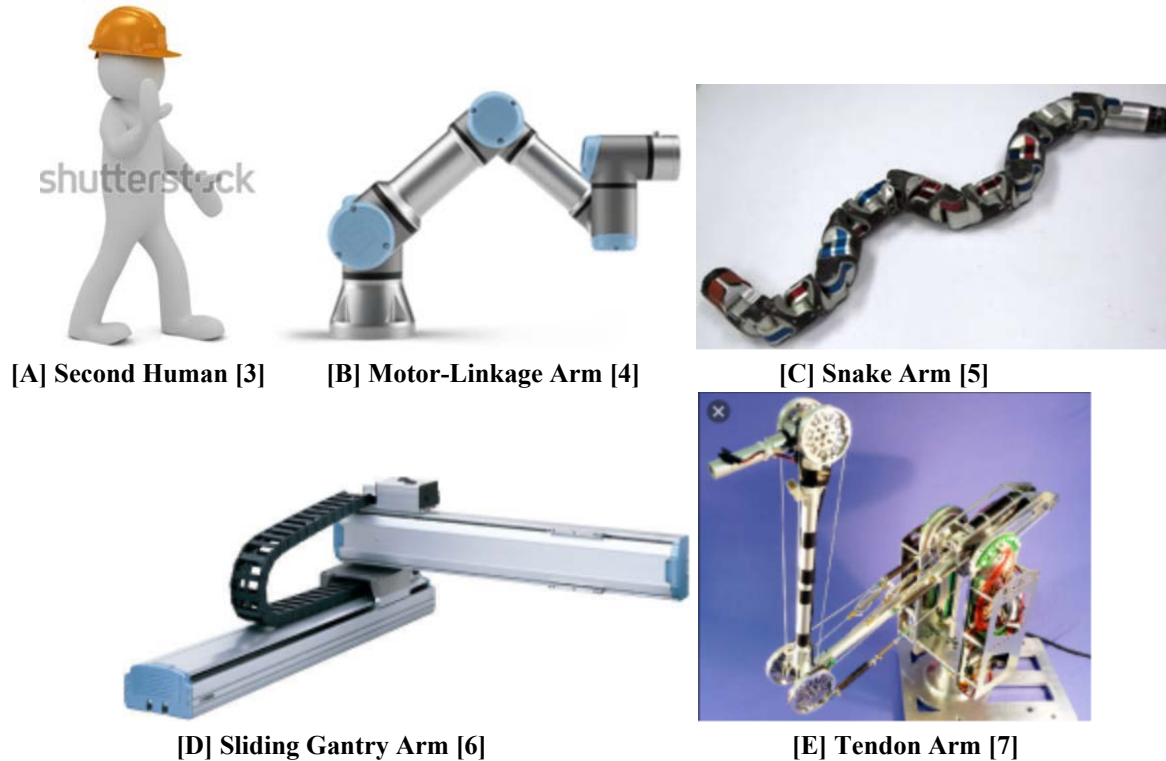
## 5. System and Subsystem-Level Trade Studies

### 5.1. System-Level

We performed a system-level trade study, as seen in Table 5 below, to determine the best mechanical framework for our task. Additionally, we compared the systems to simply having a second human as an assistant (A from Figure 3). The other options, as shown in Figure 3, include a motor-linkage arm (B), a robotic “snake” arm (C), a sliding gantry arm (D), and a tendon-actuated arm (E). The most important criteria from our trade study were the Operational Cost, Ease of Operation, Simplicity, Range, and Mechanical Advantage. The criteria for this trade study were derived from the performance and non-functional requirements. The motor-linkage arm (option B) was chosen as the best option because it provided many advantages over the other robotic options and at a much lower annual operational cost than the human alternative (option A).

Attributes			A	B	C	D	E
Upfront Cost			~\$5K	\$15K	\$50K	\$10K	\$15K
Annual Operational Cost*			\$50K	\$45	\$65	\$37	\$41
Supported Weight (lbs.)			0	30	40	50	25
DoF			Many	3	Many	2	3
Criteria	Label	Weight Factor 0-1	Value 1-10				
Low Upfront Cost	Marketing	0.3	8.4	5.8	1.8	7	6.4
Low Operational Cost	Marketing	0.9	1	9.4	7.6	9.8	8.6
Ease of Operation	NM5	0.7	8.4	7	6.2	6.4	6.2
Simplicity	NM5	0.4	9	7.4	4.2	7.2	4.6
Range	PM3.1	0.5	8.2	7	6.4	4.2	5.8
Safety	NM4	0.8	7.2	7.2	6.4	6	5
Low Maintenance Cost	Marketing	0.4	5.6	7	3.8	7.4	4.4
Aesthetically Pleasing	NM3	0.1	4.6	8	7.6	4.4	4.6
Lightweight	NM2	0.4	7.2	7.4	5.8	5.4	6
Small Form Factor	FM6	0.8	3.4	7.4	8.4	4.6	5
Success Rate	PM2	0.6	7.2	6.8	7.6	5.4	5.4
Fast Execution	PM5	0.2	9	6.8	5.6	6.8	6.2
Minimal Support Required	NM7	0.6	5.6	7.4	6	6.6	6
Mechanical Advantage	PM3.2	0.6	8.6	6.4	5.4	7.2	4.4
<b>Totals</b>			<b>6.26</b>	<b>7.33</b>	<b>6.24</b>	<b>6.48</b>	<b>5.76</b>

Table 5 - System-Level Trade Study







**Figure 3 - System-Level Trade Study Options**

## 5.2. Target Detection

We performed a trade study to determine the ideal method of part detection, as seen in Table 6 below. The options compared include detecting the user’s hands and calculating a center point, using a laser pointer to identify a location on the part [8], direct identification and detection of the part, and adding a detectable aruco marker to every part. The hand detection method was chosen as the best option because it is a reliable, hands-free method that would require minimal part manipulation. The laser pointer method was not chosen because the user would have to hold the laser, defeating the purpose of our “hands-free” device. The part detection method was not chosen because the part will be invariably occluded, which is a major problem for computer vision algorithms. Additionally, every type of part would have to be trained into the computer, which would have to reliably distinguish between them. The aruco marker method was not selected because it would require the company to add aruco markers to all of their parts that they intend to use with this device, defeating our value proposition of lowering annual costs.



Attributes			Hand Detection	Laser Pointer Detection	Part Detection	Aruco Marker Detection
Example						
Runtime complexity			Real time	Real time	Real time	Real time
Camera Requirement			3D	2D + 3D	3D, Possibly 2D + 3D	2D, Possibly 2D + 3D
Detection Accuracy			High ~90%	High ~100%	Low if occlusion	High (~97%)
Hardware Setup Difficulty			None	Low, Laser Pointer	None, Low	Medium, Attach Tags
No Occlusions			Medium	High	Low	Medium
Criteria	Label	Weight Factor 0-1	Value 1-10			
Accuracy	PM1.1	0.8	7.8	9.2	5	9.2
Low Complexity/Time Cost	PM1.2	0.7	6.4	7.6	4.4	8
Works Regardless of Occlusion	PD1.1	0.6	6.6	8.4	3.4	5.8
Works on Variety of Parts	PD1.2	0.8	9.2	9	4.4	8.4
Can Identify Pose of Part	PD1.3	0.4	2.4	1.8	8.4	8.4
Minimal Part Manipulation	NM7	0.6	9	6.2	9.2	3.8
Memory Space Usage	NM8	0.3	8	8.4	5.6	7.2
Minimal User Manipulation	NM5	1	8.5	2.5	6.75	6.5
<b>Totals</b>			<b>7.23</b>	<b>6.52</b>	<b>5.50</b>	<b>6.95</b>

**Table 6 - Part Detection Method Trade Study**

### 5.3. Hand Detection

The hand detection algorithm is the major function in the Perception Subsystem. The trade studies are split into two categories as shown in Table 7: one is the classical detection methods (the first three methods) including deep learning and machine learning algorithms, and the other is the ROS based detection packages (the last two methods).

The first three methods are the typical detection models, which output the bounding box information and the class labels. We compared two Deep Learning models with one Classical Machine Learning model. Generally, the first two deep learning methods, Mobilenet SSD [9] and YOLO [10], can run in real time, and ensure over 90% accuracy. However, the downside for YOLO is that it runs slower in CPU settings, which may make it hard to use in an industrial

setting. Both the deep learning models require fine tuning and retraining on the hand dataset. Also, the methods require ROS interfacing for this project. The Classical Machine Learning method [11], using Support Vector Machine (SVM) and bag of features, can run in real time but require more time to develop and implement.

The last two methods are about the current existing ROS implementation for hand localization. They are either open source packages or open source projects wrapped into ROS. The MIT package generates the hand cloud to localize the hands [12], and OpenPose is an open source project created by CMU, which can be ported into ROS [13]. The advantages of using these packages is that they have already been implemented into ROS, so less effort is required for us to utilize them. However, the downside is that the MIT package may need an extra calibration step every time before using it, and the OpenPose project may need a GPU to ensure low latency run time.

<b>Attributes</b>	<b>SSD</b>	<b>YOLOv4 tiny</b>	<b>SVM + Bag of Features</b>	<b>MIT ROS Package</b>	<b>OpenPose In ROS</b>
Runtime	Real time 50FPS	Real time	Real time	Real time	Real time
Accuracy	0.96	0.89	0.96	0.90	0.96
Model Complexity	Medium	Medium	Large	Small	Medium
Input Requirement	Low	Low	Medium, requires preprocessing	Medium, requires calibration	Low
Feature Engineering	No (DNN)	No (DNN)	Yes	No	No (DNN)
Hardware Requirement	No	No	No	No	No
Implementation Effort	Small	Small	Medium	Small	Medium

**Table 7 - Hand Detection Trade Study**

#### 5.4. Voice Recognition

We performed a trade study to determine the ideal voice recognition software, as seen in Table 8 below. The options compared include using Google’s API (Speech\_Recog\_UC [14]) , using the CMU-developed PocketSphinx [15], and building software from scratch. The PocketSphinx software was chosen as the best option because it is a fairly accurate package that is relatively easy to implement, and doesn’t require a connection to the internet. Speech\_Recog\_UC was not chosen because, while it’s highly accurate, and the easiest to implement, we cannot guarantee a stable Wi-Fi connection in a manufacturing environment. The build from scratch method was not selected because it would take far too much time and we could not guarantee it would have sufficient accuracy. It should be noted that, in the event of unforeseen time constraints, using Speech Recog\_UC would be entirely acceptable, but PocketSphinx is preferable.

Attributes			Speech Recog_UC	PocketSphinx	Build From Scratch
ROS Package?			ROS package	General project, but ROS supports it.	No
Runtime Complexity			Real time	Real time	Can be real time
Word Range			Large Google API word range	Medium have basic command	Can be large, or necessary words
Accuracy			Google API: 95% on word	~80% (w/ grammar auto correction)	Not sure
Keyword Trigger?			No	Yes	Yes
Internet Connection Required?			Yes	No	No
Criteria	Label	Weight Factor 0-1	Value 1-10		
Word Range	PM4.2	0.7	9.4	6.6	3.4
Accuracy of recognition	PM4.1	0.7	9	7.4	6.8
Runtime Complexity	PM1.2	0.7	8.6	8.2	7.6
Works with Variety of Users	PM4.1	0.4	9	7.4	4.6
Works with Noisy Environment	PM4.1	0.3	7.4	7.2	5.6
Memory Space	NM8	0.2	8.6	7.2	3.8
Has Keyword Trigger	NM5	0.4	4.2	9.2	8.4
Standalone (no wi-fi required)	ND1	0.5	2.2	9.6	9
Implementation Time	Schedule	0.5	8.4	6.2	2.4
<b>Totals</b>			<b>7.60</b>	<b>7.65</b>	<b>5.86</b>

**Table 8 - Voice Recognition Trade Study**

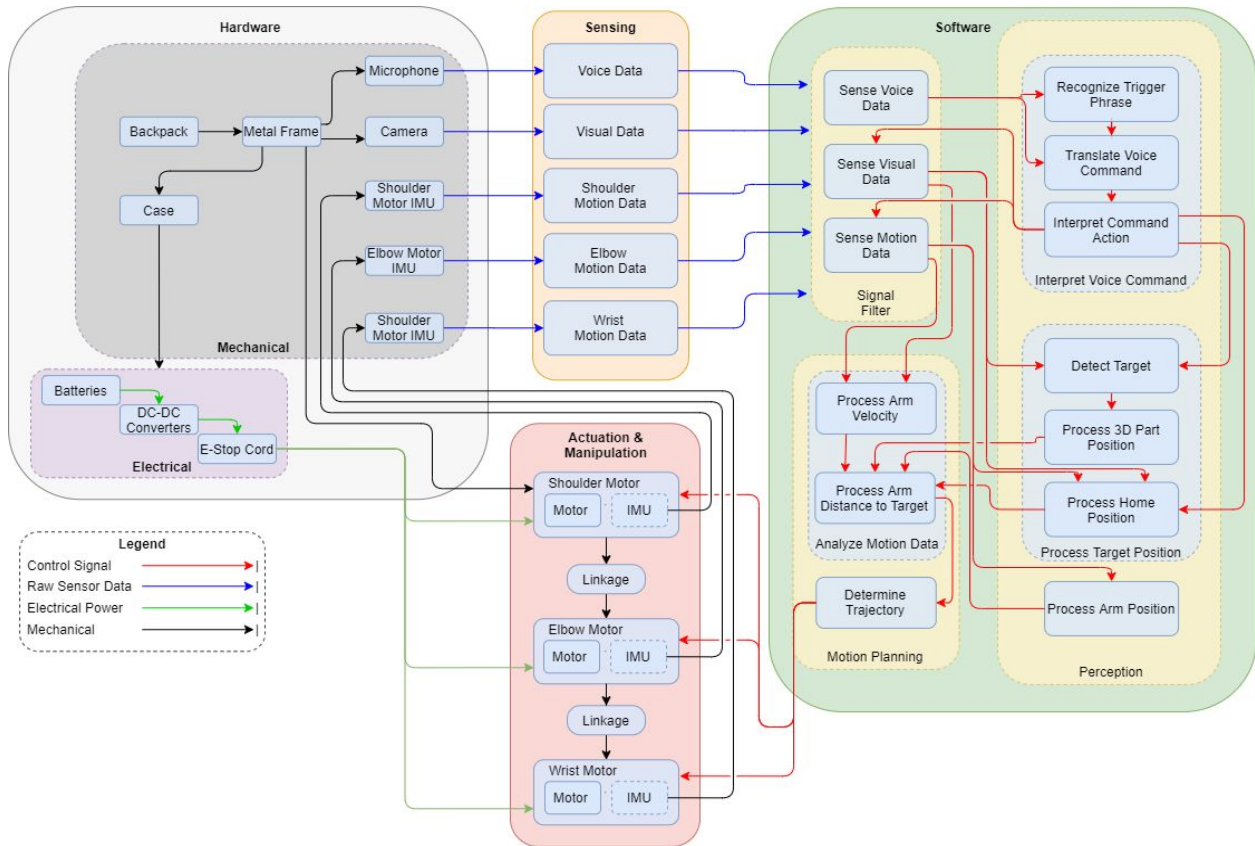
## 5.5. Actuated Manipulation - Motion Planning

We performed a trade study to determine the preferred motion planning algorithm, as seen in Table 9 below. The options included two variants of Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM). The RRT-Connect algorithm was chosen for its slightly faster reported computational runtime as well as its ability to be processed as needed, as opposed to PRM\* [16]. RRT\* was not chosen because its ability to optimize trajectory paths over time was not preferred in the use case where the user is preferring quick runtimes of the robot arm rather than smooth end effector trajectories [17]. PRM\* was not chosen because it needs to sample the environment prior to building a path and its un-optimized runtime when in a non-static environment [18].

Attributes			RRT*	RRT-Connect	PRM*
Methodology			Optimal node graphing from start to goal	Two instances of node graphing at start <b>AND</b> goal	Builds node graph among 3D environment for multiple queries
Runtime Complexity			Medium-Slow	Medium-Fast	Medium-Slow
Non-static Environment Performance			Works well	Works well	Not optimized
Target Must be in View?			Yes	Yes	Yes
ROS package?			Yes; MoveIt		
Hardware Requirements			Obstacle space, free space, target goal, kinematic constraints, dynamic constraints		
Differences			Exploring tree begins at starting position; finds viable path to target position; continues to update optimal path	Exploring tree begins at both starting and target positions; both expanding trees find connection to create viable path	Develops nodes within environment; uses graph to construct optimal path based on different target locations
Criteria	Label	Weight Factor 0-1	Value 1-10		
Smooth trajectory path	PM2	0.4	9	6	8
Minimal required data inputs	PM2	0.6	10	10	10
Small processing cost	NM8	0.6	6	7	4
Fast computational runtime	NM8	0.7	6	8	4
Processes when needed	NM8	0.6	10	10	4
Open-source in ROS	NM8	0.7	10	10	10
<b>Totals</b>			<b>8.44</b>	<b>8.67</b>	<b>6.61</b>

**Table 9 - Motion Planning Trade Study**

## 6. Cyberphysical Architecture



**Figure 4 - Cyberphysical Architecture**

The Cyberphysical Architecture shown in Figure 4 illustrates the connectivity of the entire system, mechanical to software and sensors to sensor data. The major components of this project detailed in the Functional Architecture in Figure 2 are outlined in the “Software” block of this Cyberphysical Architecture as the purpose of this project is to take an existing mechanical frame and develop autonomous capabilities for its functionality. The “Hardware” block describes the mechanical frame of the COBORG itself and the physical linkages (shown in black arrows) to the backpack, sensors, arm motors, and electrical subsystem. The “Sensing” block shows the interface between the hardware sensors to the sensor data in the software. The “Actuation & Manipulation” block details the connectivity of the motors to the hardware components and controls outputs from the “Software” block.

The “Software” block lays out the majority of the project in the autonomous software subsystems and shows the use of data from the voice and vision subsystems as well as the motion data. The voice data is sent to the “Interpret Voice Command” block to recognize the trigger word “COBORG”, then, if found, proceeds to process the rest of the voice command to determine what function to perform. The desired function is then carried out using the vision and motion data to determine the target positions, determine the arm’s current position, calculate the distance to the target, then determine the desired arm trajectory which then gets fed to the motors for actuation.

## 7. Subsystem Descriptions

### 7.1. Hardware Framework

The Hardware Framework of the COBORG is broken down into three subcomponents:

- Frame
- Case
- Manipulator Arm

The frame consists of the structural foundation of the robot that is mounted to the user. This includes the backpack that the user is wearing as well as the structural metal frame attached to the backpack straps. The manipulator arm and the case are structurally secured to the frame. See Appendix 3 for pictures of the as-built frame (A in Appendix 3).

The manipulator arm is installed to the frame and serves as the mobile unit of the robot. This includes the aluminum linkages, motors, and end effector. See Appendix 3 for pictures of the as-built manipulator arm (B in Appendix 3).

The case is the container box that houses all head-end equipment to the robot. A majority of the electrical framework components are housed in the case. Access holes are installed on the exterior of the case such that the wiring can safely travel into and out of the case. See Appendix 3 for pictures of the as-built case (C in Appendix 3).

An as-built hardware framework has already been built and tested by the sponsor. Revisions to the Hardware Framework shall refer to the as-built counterpart as the fallback design.

### 7.2. Electrical Framework

The Electrical Framework consists of the electrical components and wiring that are installed in the COBORG. The following electrical components shall be housed in the case of the hardware framework:

- Router and wireless antennas
- Ruggedized portable batteries
- Power distribution board
- Workstation computer

Power and data cabling travel into and out of the case. Cables are strung around structural members and the frame to prevent slack and hanging cables during usage.

A baseline Electrical Framework has already been built and tested by the sponsor. Revisions to the Electrical Framework shall refer to its as-built counterpart as the fallback design.

### 7.3. Sensors

The following sensors shall be installed and used on the COBORG:

- Intel Realsense D435 depth camera (1)
- Intel Realsense T265 localization camera (1)
- IMU sensor (internal to each HEBI motor) (3)
- Microphone (1)

The Realsense cameras and microphone shall be mounted onto the frame and in close proximity to the user's shoulder(s). IMU sensors exist internally in each of the three HEBI motor models being used on the COBORG.

The two camera models (D435 and T265) are common models in the biorobotics lab and are inexpensive to purchase. Given the common usage and confidence of the two camera models in the biorobotics lab, no fallback/alternative model has been chosen.

The HEBI motor models are common models used in the biorobotics lab and several spare units exist in the lab space. Given the common usage and confidence in the HEBI motors in the biorobotics lab, no fallback/alternative model has been chosen.

Additional trade studies and analysis are recommended but not required to select a microphone model and a fallback/alternative microphone model.

### 7.4. Perception: Vision

The vision subsystem consists of the target detection algorithm of the intended part. In order to detect the target location for the robot arm to reach, a hand location (which is holding the target) can be used to indicate the target location. The bounding box of the user's hand is detected when the user says "COBORG, hold this." From the bounding box, a location for the target location is determined based on the edge of the user's hand which is holding the part for the robot arm to reach.

The current choice for the hand detection algorithm is the open-source algorithm YOLOv4. The algorithm already has an implementation for hand detection that can be interfaced with ROS.

The fallback/alternative model would be through an open-source ROS package for hand detection such as the MIT Hand Point Cloud or OpenPose in ROS package. Refer to Table 7 for trade study comparisons between different hand detection models.

### 7.5. Perception: Voice

The voice subsystem will receive the user's voice audio and outputs, then determine the state change signals for the COBORG. The voice algorithm shall analyze the audio input of the user and compare those input signals with a database of predefined commands. Once the voice algorithm determines the intended predefined command from the audio input, the subsystem shall output the appropriate state change signal for the other subsystems to act appropriately.

The current selection for the voice algorithm is the CMU PocketSphinx package that can be interfaced with ROS.

The fallback/alternative choice is using Google's Voice API that already has ROS implementations. Refer to Table 8 for trade study comparisons between different voice recognition package options.

## 7.6. Actuated Manipulation

The actuated manipulation subsystem consists of two components:

- Motion planning model
- Trajectory controller

The motion planning component of the actuated manipulation subsystem is the algorithm that is responsible for planning a viable trajectory path from the current position of the manipulator arm to the determined target location in 3D space. Most commonly, the manipulator arm will be planning a path from the compact/home position.

The trajectory controller component of the actuated manipulation subsystem is the lower-level controller that controls the motors of the robot arm through multiple states. When the robot arm reaches the target location, the trajectory controller is in charge of controlling the joint angles and torques of each motor through each waypoint. When the robot arm stabilizes the target part in 3D space, the trajectory controller shall receive inputs from the IMU sensors to control the joint angles and torques on the motors.

For the motion planning components, the current algorithm selected is RRT-Connect. The fallback/alternative algorithm would be another variant of RRT, such as RRT\*. Refer to Table 9 for trade study comparisons between the different motion planning models.

For the trajectory controller, the current model selection would be the native joint angle controller and torque limit declarations derived from the MATLAB HEBI implementation. The trajectory controller was developed and tested by the Biorobotics lab. Any changes to the model of the trajectory controller will refer to the as-built Biorobotics implementation as the fallback/alternative model.

## 8. Project Management

### 8.1. Work Plan and Tasks

Figure 5 depicts the high level Work Breakdown Structure (WBS) required to execute the COBORG robot system. From the high level breakdown we were able to derive the tasks needed to accomplish each of the work packets detailed in the WBS. These tasks are detailed in Table 10 which is a tabular form of the lower level WBS:



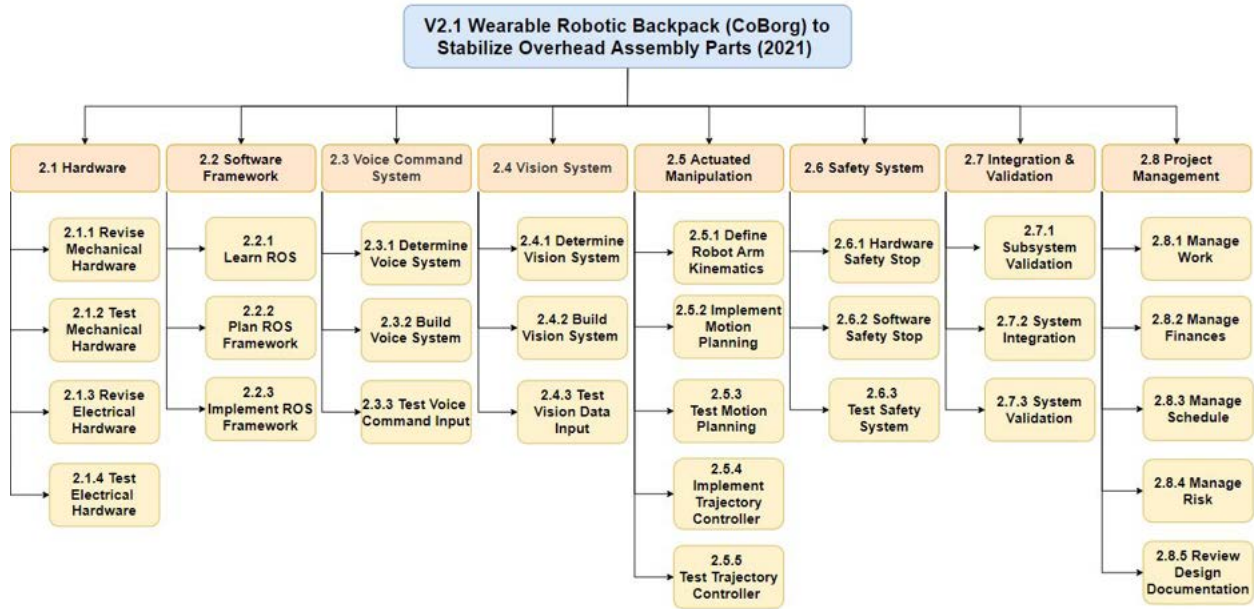


Figure 5 - WBS Structure Level 3

Subsystem	Subsystem Work Breakdown	
2.1 Hardware	2.1.1 Revise Mechanical Hardware	2.1.1.1 Create BOM for As-Is Build
		2.1.1.2 Create 3D Printed Adapters to Hold Sensors
		2.1.1.3 Design and Create Housing for COBORG
		2.1.1.4 Sleeve Robot Arm
	2.1.2 Test Mechanical Hardware	
	2.1.3 Revise Electrical Hardware	2.1.3.1 Create Electrical Schematics for As-Is Build
2.1.3.2 Create PCB to Replace Disjointed Components		
2.1.3.3 Create Hot Swappable Power System		
2.1.4 Test Electrical Hardware		
2.2 Software Framework	2.2.1 Learn ROS	2.2.1.1 Complete “ROS Basics in 5 Days” Module
	2.2.2 Plan ROS Framework	2.2.2.1 Create ROS Framework Node and Topic Map
		2.2.3 Implement ROS Framework
	2.2.3.2 Create Voice System Node	
	2.2.3.3 Create Actuated Manipulation System Node	
2.3 Voice Command System	2.3.1 Determine Voice System	2.3.1.1 Create Voice Program Trade Study
		2.3.1.2 Select Voice System
	2.3.2 Build Voice System	2.3.2.1 Preprocess Audio Data
		2.3.2.2 Implement Keyword Recognition Algorithm

		2.3.2.3 Implement Grammar Correction Algorithm
		2.3.2.4 Integrate Keyword and Grammar Checking System
	2.3.3 Test Voice Command Input	
2.4 Vision System	2.4.1 Determine Vision System	2.4.1.1 Create Vision Trade Study
		2.4.1.2 Select Vision System
	2.4.2 Build Vision System	2.4.2.1 Obtain Raw Image and Point Cloud Data
		2.4.2.2 Implement Object Detection Algorithm in ROS
		2.4.2.3 Implement 3D Localization Algorithm of Hand
		2.4.2.4 Implement 3D Localization of Moving Camera Frame
2.4.3 Test Vision Data Input		
2.5 Actuated Manipulation	2.5.1 Define Robot Arm Kinematics	2.5.1.1 Define Joint and Frames Using DH Parameters
		2.5.1.2 Build URDF Model
	2.5.2 Implement Motion Planning	2.5.2.1 Select Best Motion Planning Algorithm
		2.5.2.2 Implement Motion Planner Package
	2.5.3 Test Motion Planning	
	2.5.4 Implement Trajectory Controller	2.5.4.1 Implement Motor API Package
		2.5.4.2 Create Basic Joint Angle Constraints
		2.5.4.3 Implement Stabilization Algorithm
		2.5.4.4 Introduce Advanced Trajectory Constraints Around User
	2.5.5 Test Trajectory Controller	
2.6 Safety System	2.6.1 Hardware Safety Stop	2.6.1.1 Validate Current Push Button EStop
		2.6.1.2 Upgrade EStop from Push Button to Pull Cord
	2.6.2 Software Safety Stop	2.6.2.1 Implement Voice Command Safety Stop
		2.6.2.2 Implement Human Collision Avoidance
	2.6.3 Test Safety System	
2.7 Integration & Validation	2.7.1. Subsystem Validation	2.7.1.1 Hardware Validation
		2.7.1.2 Software Framework Validation
		2.7.1.3 Voice System Validation
		2.7.1.4 Vision System Validation
		2.7.1.5 Actuated Manipulation Validation
		2.7.1.6 Safety System Validation
	2.7.2 System Integration	2.7.2.1 Hardware Integration
		2.7.2.2 Software Framework Integration
		2.7.2.3 Voice System Integration
		2.7.2.4 Vision System Integration

	2.7.3 System Validation	2.7.2.5 Actuated Manipulation Integration
		2.7.2.6 Enhanced Safety Integration
		2.7.3.1 Internal System Demo 1
		2.7.3.2 Internal System Demo 2
2.8 Project Management		2.7.3.3 Fall Demonstration
		2.8.1 Manage Work
		2.8.2 Manage Finances
		2.8.3 Manage Schedule
		2.8.4 Manage Risk
		2.8.5 Review Design Documentation

**Table 10 - WBS Structure Level 4**

From these work packets we were able to understand our timeline in regards to our schedule and when key milestones should take place.

## 8.2. Key Milestones & Schedule

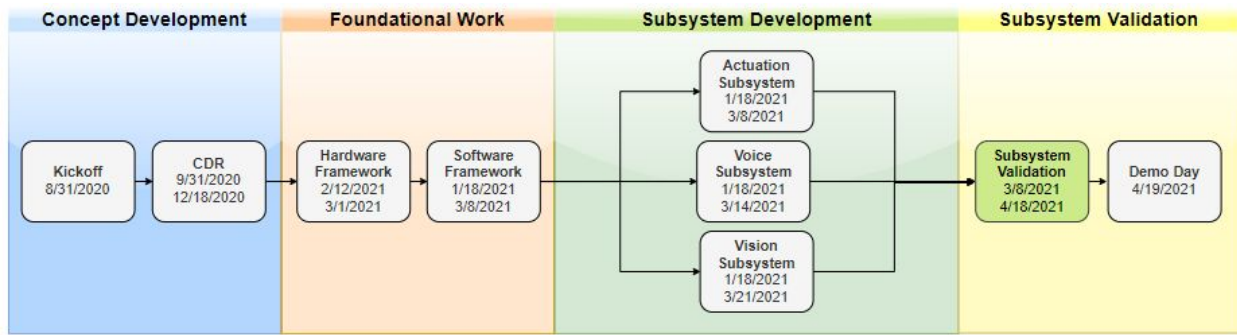
The Spring Key Milestones are shown below in Table 11. The external milestone for spring is the “Spring Validation Demonstration” (SVD), which is on April 19th. Before the SVD, five internal milestones, including Subsystem Completion and Subsystem Internal Validation, will take place between February and early April.

The Fall high-level milestones are shown in Table 17 in Appendix 1. In the Fall, we will finish the subsystem integration (estimated by the end of September) and the added feature integration (estimated by the end of October) before doing the Fall Validation Demonstration (in the end of November).

<b>Date</b>	<b>Milestones</b>
Feb. 20	Hardware/Sensing Subsystem Launch
<b>Mar. 8</b>	<b>Progress Review 1</b>
Mar. 14	Actuated Manipulation Subsystem Completion
Mar. 21	Voice Subsystem Completion
<b>Mar. 22</b>	<b>Progress Review 2</b>
Mar. 28	Vision Subsystem Completion
Apr. 5	Subsystem Internal Validation
<b>Apr. 19</b>	<b>Spring Validation Demonstration</b>

**Table 11 - Key Milestones for Spring**

Our plans for progress reviews are to demonstrate the functionality of the subsystems as they are developed in the Spring. Figure 6 and Gantt Chart in Appendix 2 details our execution timeline and provides the estimated dates of completion for the Vision, Voice, and Actuation Subsystems of the COBORG platform:



**Figure 6 - Spring Schedule Diagram**

From this schedule we estimate to have completed the base of the software framework by Progress Review 1 on March 8th, and by Progress Review 2 on March 22nd we will have basic functionality of major subsystems. With this in mind, our plan for the progress reviews are as follows:

<b>Progress Review 1</b>	Demonstrate basic hardware functionality (Motor Movement)
	Show transfer of data between subsystem nodes (Print/Toggle test)
<b>Progress Review 2</b>	Show output of data from 3D camera components (Vision)
	Show output of data from microphone (Voice)
	Show RVIZ demonstration of robot arm movement (Actuation)

**Table 12 - Progress Review Demonstrations**

### 8.3. System Validation Experiments

#### 8.3.1. Spring Validation Demonstration

The Spring Validation Demonstration will be held in the Biorobotics Lab in the basement of Newell-Simon Hall. The necessary equipment includes the COBORG backpack, the vision subsystem, the voice subsystem, a computer to run the subsystems, a stopwatch, a ruler, the record sheets, a measuring tape, various set-up stands, and a plate with shiny and matte finishes on either side. An area of roughly 10 feet in diameter will be required for the test.

First, the vision subsystem will be tested. The plate and camera will be fixed at specific locations, and their relative pose measured. Then, a team member will place their hands on the plate and the camera will be signaled to determine its pose. The estimated pose will be compared to the ground truth, validating PM1.1, PD1.1, and PD1.3. The time from signalling to pose estimation result will be recorded, validating PM1.2. The plate will then be flipped to its other side and the test run again, validating PD1.2. This test will be repeated in two other poses to build confidence in the validation.

Second, the actuation and manipulation subsystem will be tested. The COBORG backpack and a target will be fixed at specific locations, and their relative pose measured. Then, the backpack will be signaled to reach out to the target. The distance between the target and the center of the paddle will be measured, validating PM2. Then, the COBORG backpack will be

signaled to return to compact mode. The maximum distance from the center of the first motor to any distal part of the arm will be measured, validating PM6. This process will be repeated. Then the process as a whole will be repeated in two different relative poses to build confidence in the validation. Afterwards, the COBORG backpack arm will be extended at a shallow, downward angle and a 2 lb weight will be attached to the end. The COBORG backpack arm will be signaled to lift the weight up beyond horizontal, validating PM3.2.

Third, the voice subsystem will be tested. During the test, every trigger of the microphone will be recorded, as well as triggers that were supposed to occur, but did not. The microphone will be signaled to start listening. The user will speak command #1 and record the subsystem's response. The user will then read a short paragraph. The user will then speak command #2 and record the subsystem's response. The user will then read another short paragraph. This process will be repeated twice to build confidence in the validation. Comparing the number of correct and incorrect responses versus triggers will validate PM4.1. Using appropriate commands will validate PM4.2 and PM4.3.

Finally, the nonfunctional aspects of the system will be tested. The COBORG backpack will be weighed on a scale, validating NM2. The COBORG backpack will be put on by all members present and their remarks about its comfort will be recorded, validating NM1.

### 8.3.2. Fall Validation Demonstration

The Fall Validation Demonstration will be held in the Snake Robotics Lab in the basement of Newell-Simon Hall. The necessary equipment includes the fully-integrated COBORG backpack, a stopwatch, a ruler, the record sheet, a measuring tape, a power drill, 8 screws, a scaffolding "airplane wing" set up, and a plate with shiny and matte finishes on either side. An area of roughly 10 feet in diameter will be required for the test.

The entire COBORG backpack system will be tested with a simulated use case scenario. A team member will put on the backpack and turn it on. They will then pick up the plate and hold it in its proper position for attachment. The team member will command the COBORG backpack to hold the part. The COBORG backpack arm will extend out to the part to secure it. The COBORG backpack executing the appropriate action will validate PM4.1, PM4.2, and PM4.3. The time between the voice command and action initialization will be recorded, validating PM1.2. The team member will then remove their hands and attach the part using the hand drill and screws. The robot's success in securing the part will validate PM1.1, PM2, PM3.1, PD1.1, NM8, and ND1. The team member will then command the COBORG backpack to enter compact mode. The COBORG backpack arm will fold itself up into a predetermined pose. The time between the voice command and action initialization will be recorded, validating PM5. The maximum distance from the center of the first motor to any distal part of the arm will be measured, validating PM6. The process will be repeated in order to remove the screws from the plate and take it down. The plate will then be flipped over to an alternate-textured side and the whole process will be repeated, validating PD1.2 and NM7. The entire process up to this point will be repeated two more times to build confidence in the validation. Injury-free success will validate NM4. A lack of problems with the system will validate NM5. The battery lasting throughout the entire trial will validate NM6.

#### 8.4. Team Member Responsibilities

To reduce the risk of a subsystem being the responsibility of one team member, each team member will have a secondary responsibility. The subsystems which require 100% overlap are the Vision, Voice, and Actuated Manipulation portions of the project, as they will be built ground up without prior implementation. The structure of the overlap was determined by the members primary responsibility, their interests, and Spring semester workload. This is detailed in the table below:

<b>Team Member</b>	<b>Primary Responsibility</b>	<b>Secondary Responsibility</b>
Feng Xiang	Actuated Manipulation	Vision Subsystem
Yuqing Qin	Vision Subsystem	Voice Subsystem
Gerry D'Ascoli	Voice Subsystem	Electrical Hardware
Jonathan Lord-Fonda	Integration Validation	Mechanical Hardware
Husam Wadi	Project Management	Software Framework

**Table 13 - Team members and responsibilities**

#### 8.5. Parts List and Budget

The parts list detailed in Table 14 is preliminary and covers spares and necessary components for the Spring semester. Our goal is to retain a majority of the project funds (\$4,000) to allow for hardware expansion in the Fall. While we are budgeting for a Hebi Motor Module, we will not purchase the part until if/after a motor failure occurs and have accounted for this in our risk analysis. We will focus primarily on software related expenses for the SVD, and the parts required for subsystem demonstration:

<b>No.</b>	<b>Part Name</b>	<b>Cost</b>	<b>Quantity</b>	<b>Total Cost</b>
1	ROS Training Courses	\$59.95	5	\$299.75
2	3D Printed EOAT Spare	\$89.99	1	\$89.99
3	USB 3.1 M-F adapter	\$5.37	2	\$10.74
4	Hebi Motor Module	\$3310.51	1	\$3310.51
<b>Total</b>				<b>\$3710.99</b>

**Table 14 - Part List and Budget**

#### 8.6. Risk Management

An important aspect of every project is taking into account potential risks that may arise as the project progresses. These risks span all aspects of the project, from technical to programmatic, and from inconsequential to “show-stopping”. To ensure that our project progresses successfully, we have taken into account the COBORG project risks and detailed tasks taken to mitigate the severity or likelihood of these risks occurring:

<b>Risk</b>	<b>Label</b>	<b>Mitigation Plans</b>
Hebi motor module dies	RT1	Ensure at least one spare is secured.
		Reduce budget to \$2,000 to afford spare
Main computer dies	RT2	Work out of Cloud, Github Repository
		Ensure spare workstation is available
		Budget \$1300 to purchase spare (Zotac)
Estop devices malfunction	RT3	Inquire Biorobotics lab for spare component
		Inquire John's inventory for spare device(s)
		Ensure Estop device is fail-open
TCP/IP connectivity is lost	RT4	Inquire Biorobotics lab for spare
		Inquire John's inventory for spare device(s)
		If no spare, allocate funds to purchase spare
Water damage to COBORG System	RT5	Seal backpack and perform liquid spill test
		If operating outside, ensure weather is favorable
		Ensure all liquids are 6ft from robot exoskeleton
Team lacks ROS fundamentals by start of spring semester	RR1	Execute plan to learn ROS over winter break
		Enforce one week boot camp before Spring
		Hire ROS SME to build framework for project
Unable to work 10hrs/week/member on MRSD project	RR2	Offload work amongst team in certain situations
		Treat MRSD project deadlines as HW deadlines
Hazard occurs on user while wearing robot	RR3	Be prepared to act when the issue arises
		Brief and prep the user on proper procedure
		Perform on-the-rack run through before testing
		Set torque limits and vision obstacle avoidance
Member contracts COVID	RP1	Follow CMU pandemic safety procedures
		Create simulation of system in ROS
MRSD program gets disrupted due to COVID pandemic	RP2	House robot outside of lab environment
		Create simulation of system in ROS
Our sponsor graduates in the spring of 2020	RP3	Confirm Julian can commit time after graduation
		Acquire non-CMU contact information

**Table 15 - Top 11 Risks Mitigation Plans**

By taking into consideration these risks, which span technical, resource, and programmatic concerns, we have ensured that none of the risks found have critical impact or likelihood to the project as it is executed. This is detailed in the risk table below:

<b>Risk</b>	<b>Label</b>	<b>Category</b>	<b>Mitigated Likelihood</b>	<b>Mitigated Consequence</b>	<b>Risk Level</b>
Hebi motor module dies	RT1	Technical	3	2	Low
Main computer dies	RT2	Technical	1	4	Low
Estop devices malfunction	RT3	Technical	2	3	Low
TCP/IP connectivity is lost	RT4	Technical	4	2	Medium
Water damage to COBORG System	RT5	Technical	1	4	Low
Team lacks ROS fundamentals by start of spring semester	RR1	Resource	3	2	Low
Unable to work 10hrs/week/member on MRSD project	RR2	Resource	4	2	Low
Hazard occurs on user while wearing robot	RR3	Resource	2	4	Medium
Member contracts COVID	RP1	Programmatic	3	2	Low
MRSD program gets disrupted due to COVID pandemic	RP2	Programmatic	3	3	Medium
Our sponsor graduates in the spring of 2020	RP3	Programmatic	5	1	Low

**Table 16 - Top 11 Mitigated Risks Table**



## 9. References

- [1] Owano, N. (2018, June 21). Ford, exoskeleton company address strain in overhead tasks. Retrieved December 18, 2020, from <https://techxplore.com/news/2018-06-ford-exoskeleton-company-strain-overhead.html>
- [2] Farris, Riley & Pitt, LLP. (n.d.). Retrieved December 18, 2020, from <https://www.frplegal.com/industrial-accidents/8-most-common-workplace-injuries-for-factory-workers/>
- [3] Trace, N. (n.d.). 3D man in helmet (Little Human Character) shows a hand that everyone needs to stop. Retrieved December 18, 2020, from <https://www.shutterstock.com/image-illustration/3d-man-helmet-little-human-character-113274703>
- [4] UR3e - Universal Robots. (n.d.). Retrieved December 18, 2020, from <https://www.ebay.com/itm/UR3e-Universal-Robots-/202820281210>
- [5] P. (2014, October 12). Snake robot to aid search-and-rescue operations. Retrieved December 18, 2020, from <https://www.indiatoday.in/science-and-technology/story/snake-robot-search-and-rescue-operation-carnegie-mellon-university-sidewinders-science-222832-2014-10-12>
- [6] (n.d.). Retrieved December 18, 2020, from <https://junceraautomations.com/yamaha-robotics>
- [7] Lens, T., & Stryk, O. (1970, January 01). Figure 1 from Design and dynamics model of a lightweight series elastic tendon-driven robot arm: Semantic Scholar. Retrieved December 18, 2020, from <https://www.semanticscholar.org/paper/Design-and-dynamics-model-of-a-lightweight-series-Lens-Stryk/4e34fec29247ed9e9e361010deef5abbe5afd8d/figure/0>
- [8] GlobalToronto. (2019, January 25). Panasonic unveils "robot arm" prosthetic. Retrieved December 18, 2020, from <https://www.youtube.com/watch?v=0ilWLfs1PKQ>
- [9] Yangli18. (n.d.). Hand\_detection. Retrieved December 18, 2020, from [https://github.com/yangli18/hand\\_detection](https://github.com/yangli18/hand_detection)
- [10] Bruggisser, F. (n.d.). Yolo-hand-detection. Retrieved December 18, 2020, from <https://github.com/cansik/yolo-hand-detection>
- [11] Dardas, N. H., & Georganas, N. D. (2011). Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(11), 3592-3607. Retrieved December 18, 2020 from <https://doi.org/10.1109/TIM.2011.2161140>
- [12] Gallagher, G. (n.d.). Hand-Detector. Retrieved December 18, 2020, from <http://wiki.ros.org/mit-ros-pkg>

- [13] Anush19o5. (n.d.). Openpose\_ros. Retrieved December 18, 2020, from [https://github.com/anush19o5/openpose\\_ros](https://github.com/anush19o5/openpose_ros)
- [14] Oliveira, J. P., & Perdigao, F. (n.d.). Speech Recognition Package. Retrieved December 18, 2020, from [https://github.com/jopedroliveira/speech\\_recog\\_uc](https://github.com/jopedroliveira/speech_recog_uc)
- [15] Humpelstilzchen. (2018, December 08). Speech recognition in ROS with PocketSphinx. Retrieved December 18, 2020, from <https://hackaday.io/project/25406-wild-thumper-based-ros-robot/log/156823-speech-recognition-in-ros-with-pocketsphinx>
- [16] Kuffner, J. J., & LaValle, S. M. (2000, April). RRT-connect: An efficient approach to single-query path planning. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, 00CH37065(2)*, 995-1001. Retrieved December 18, 2020 from <https://doi.org/10.1109/ROBOT.2000.844730>
- [17] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7), 846-894. Retrieved December 18, 2020 from <https://arxiv.org/abs/1105.1186>
- [18] Marble, J. D., & Bekris, K. E. (2013). Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics*, 29(2), 432-444. Retrieved December 18, 2020 from <https://doi.org/10.1109/TRO.2012.2234312>

# 10. Appendix

## Appendix 1 - Key Milestones for Fall 2021

Date	Milestones
Sep. 20	Subsystem Integration Completion
Oct. 18	Added Tasks Completion
Nov. 2	Full System Integration Internal Validation
Nov. 15	Fall Validation Demonstration

Table 17 - Key Milestones for Fall Semester

## Appendix 2 - Gantt Chart for Spring Semester

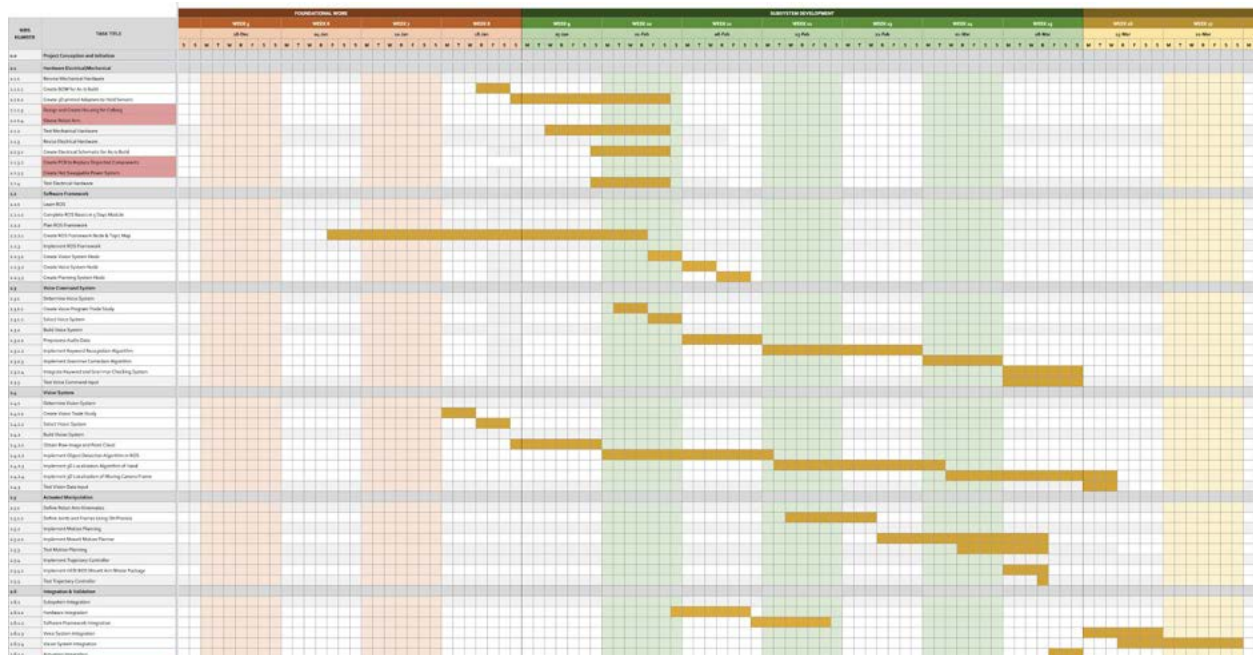


Figure 7 - Gantt Chart for Spring Semester Detailed Schedule

Appendix 3 - Pictures of As-Built COBORG



(A) Front View

(B) Side View

(C) Rear View

Figure 8 - As-Built COBORG Framework