University of Colorado Boulder

ASEN 4028 Senior Projects II: Design Practicum

RIVeR

May 3, 2021

Authors:

Project Manager: Kyle Li Email: kyli8681@colorado.edu	Lead Systems Engineer: Lindsay Cobb Email: lindsay.cobb@colorado.edu Phone: 415- 450-7535
Name: Logan Vangyia Email: lova2860@colorado.edu Position: Translator Lead	Name: Nicholas Miller Email: nimi1223@colorado.edu Position: End Effector Lead
Name: Marin Grgas Email: magr4013@colorado.com Position: Structures/Manufacturing Lead	Name: James Tiver Email: Jati6685@colorado.edu Position: Software Lead
Name: Jordan Abell Email: joab9590@colorado.edu Position: Software/Dynamics Lead	Name: Bruce Barnstable Email: brba0083@colorado.edu Position: Safety Lead
Name: Jett Moore Email: jemo0269@colorado.edu Position: Electrical Lead	Name: Brandon Torczynski Email: brto2556@colorado.edu Position: Testing Lead
Name: Alex Ferguson Email: alfe2986@colorado.edu Position: Finance Lead	Name: Peter Amorese Email: peam6502@colorado.edu Position: Robotic Arm Lead

Advisor: Dr. Neogi

Customer: Sierra Nevada Corportation and Loren McDaniel

Abstract

Contents

List of Figures

List of Tables

List of Acronyms

IVR	Intra-Vehicular Robotics
ISS	International Space Station
LIFE	Large Inflatable Fabric Environment
SNC	Sierra Nevada Corporation
ROS	Robotic Operating System

Nomenclature

a	acceleration
dy	Offset in the y-direction
dz	Offset in the z-direction
δ	Deflection
E	Young's Modulus
F	Force
F_a	Axial Force
$\mathcal{H}(\hat{x})$	Nonlinear Sensor Measurement Function
J(t)	Cost Function
k	Measurement Time Step
T	Torque
η	ball screw efficiency
l	lead screw length
I	Second Moment of Inertia
M	Moment
m	Mass
w	Weight
R	Lever Arm
r	Distance Vector
x	Distance
y(k)	Sensor Measurement at Time k
$\bar{y}$	Y Centroid
$\bar{z}$	Z Centroid
$\hat{y}$(k)	Predicted Sensor Measurement at Time k

1 Project Purpose

Authors: Lindsay Cobb

1.1 Mission Context

The Sierra Nevada Corporation has designed the LIFE Habitat (Large Inflatable Fabric Environment) for Phase 3 of NASA's Next Space Technologies for Exploration Partnerships (NextSTEP-2). This module is designed to be launched on a rocket and inflate once in orbit to house a crew of 4 for long duration missions *. NASA has outlined requirements for an Intra-Vehicular Robotics (IVR) systems that is capable of performing inspection, maintenance, repair, housekeeping, payload operations, and logistics/cargo management during autonomous (un-crewed) operations within the pressurized ambient environment of the habitat.

Project RIVeR is designed to function as an IVR system within the LIFE habitat and focus on cargo management and distribution. RIVeR will demonstrate a proof of concept that robotics are capable of performing of tasks required on an uncrewed space habitat. RIVeR will begin operations once a supply vehicle has docked with the cargo hatch and a separate IVR system has positioned the cargo bags at the opening of the hatch.

1.2 Project Purpose

Mission Statement: RIVeR will prove the feasibility of using Intra-Vehicular Robotics (IVR) to identify and distribute cargo bags within Sierra Nevada's LIFE™ Habitat to demonstrate task management in an uncrewed environment.

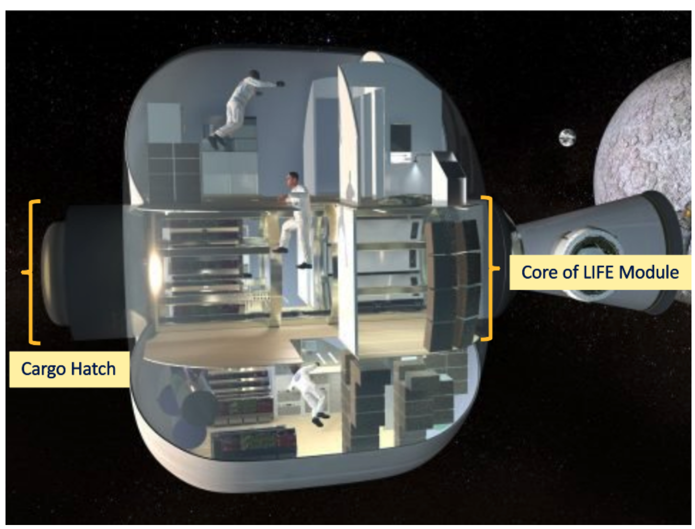


Figure 1: Habitat Environment

*Schwandt, Kimberly. "Ozmens' SNC Advances Habitat Development for NASA". Sierra Nevada Corporation

RIVeR will focus on the mid-level, or Core, operations of the LIFE habitat for cargo bag identification and distribution. RIVeR will use an imaging sensor system to identify cargo bags that are positioned in the opening of the cargo hatch. A robotic arm will retrieve the bags from the hatch and deposit them at a drop off location.

1.3 Application

Currently, supplies for the astronauts on the International Space Station (ISS) arrive on cargo vehicles which are docked to the station. Astronauts spend hours unloading and transporting the cargo throughout the ISS and securing it in one of the modules. The LIFE habitat is intended to be completely outfitted with supplies when the astronauts arrive in orbit. This may require several cargo resupply missions to fully stock the module. The IVR system's first task will be unloading cargo and distributing it to different levels of the LIFE module. RIVeR will be a singular step in the entire process and is the first proof of concept demonstration for this application.

2 Project Objectives and Functional Requirements

Authors: Lindsay Cobb

2.1 Levels of Success

Level	Translator	Robotic Arm	End Effector
Level 1	Design a platform that is capable of being mounted to the rail system of size 1.6m x 0.3m.	Robotic arm can move to a desired pose under a given command without colliding with simulated LIFE module environment.	End effector is able to take a command to operate the bag capture mechanism.
Level 2	Translator is able to integrate with the robotic arm including power and communication systems.	Robotic arm can plan and move to a specified pose while the base is being moved by the translator.	End effector can capture bag with operator input and maintain hold while translating and rotating the arm.
Level 3	Translate robotic arm up to 1.6 meters in one direction given a control input with 1 cm of accuracy.	Robotic system can capture a bag and release it at a specified location, with a remote operator determining pick up and drop off location.	End effector receives input from the robotic arm to be aligned, capture, and control a bag instead of a remote operator.
Level 4	Translation is automated and repeatable; sensor suite returns position data to the system/user to refine position during operations.	The system will complete a cargo transportation task by identifying, locating, capturing, and releasing a bag with no manual inputs from an operator.	The end effector is correctly aligned to capture a bag based on the coordinate location returned by the imaging sensors.

Table 1: Levels of Success

2.2 Concept of Operation

The Concept of Operations for project RIVeR is separated into three stages; Capture, Translation, and Release. These steps will be executed to identify a cargo bag, capture, and then release it at a new location.

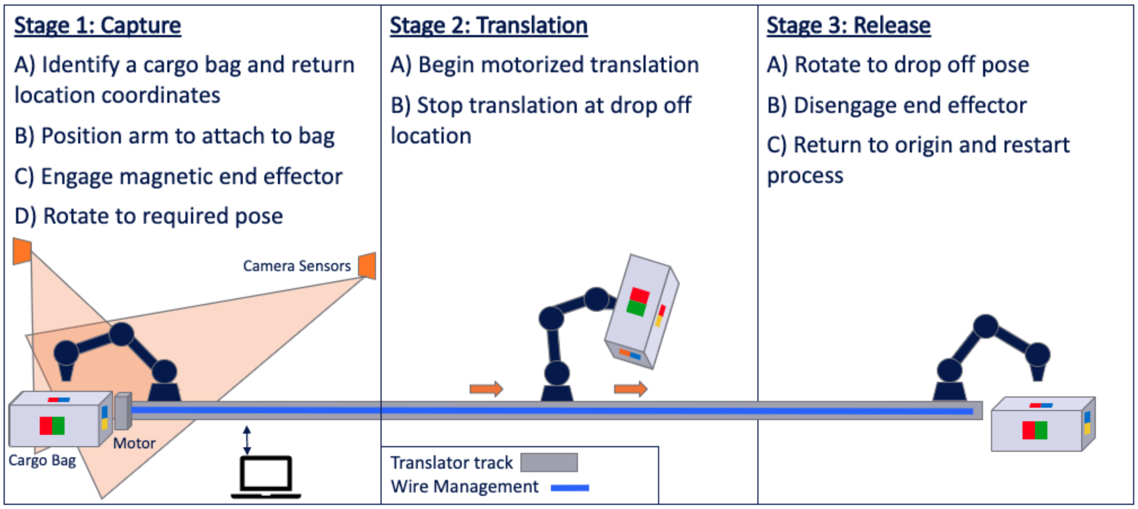


Figure 2: Concept of Operation

2.3 Functional Block Diagram

The functional block diagram displays all of the subsystems and how they interface with each other. Likewise, data rates shown between the various electronic components and the power supplied to each. The legend in the bottom, right shows the types of connections between the various parts.

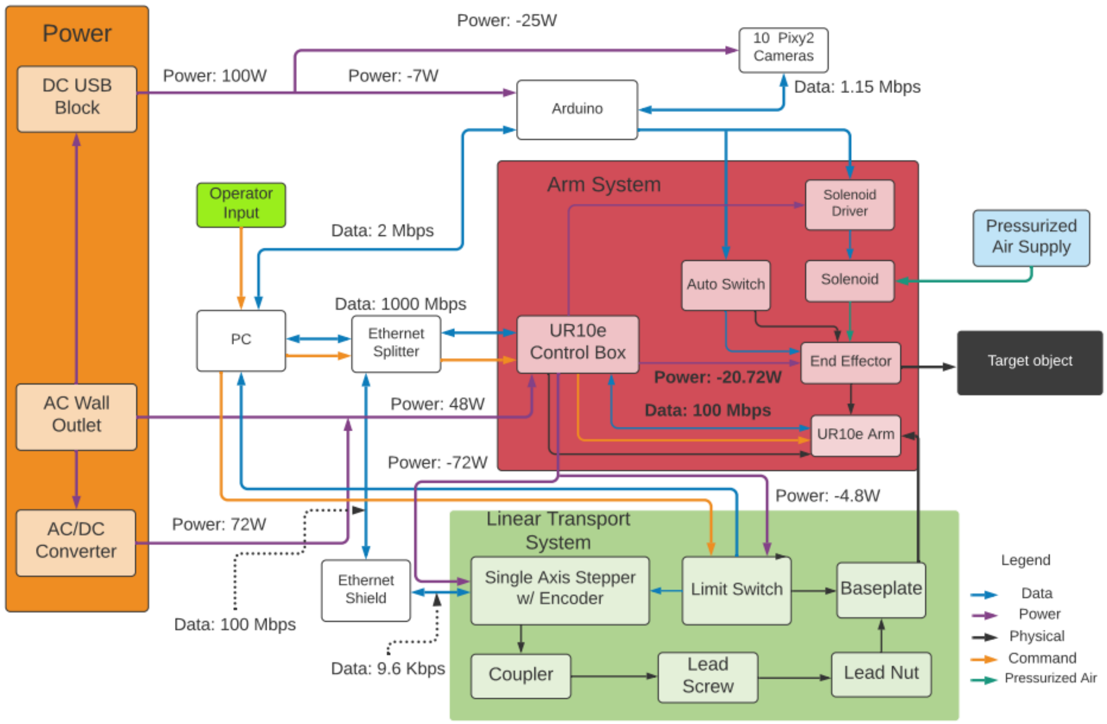


Figure 3: Functional Block Diagram

2.4 Functional Requirements

FR.1 All systems shall be operational in a 1G testing environment.

Source: Derived from Customer Requirement 3.1 listed in Appendix **??**

Rationale: While the project is designed to operate in a zero gravity environment, the system must be capable of supporting its own weight so it can be tested on Earth.

FR.2 The system shall be capable of translating a cargo bag the length of the track.

Source: Derived from Project Purpose

Rationale: The track is representative of the distance that cargo will need to be transported. The anticipated complications that will need to be addressed with regards to track length are wire management, motor calibration, and image recognition.

FR.3 The system shall be capable of determining the status of a cargo transportation task.

Source: Derivative of NASA Requirement L3-HAB-0145

Rationale: The system must be capable of identifying where a cargo bag is located during the entire operation including whether or not it is irretrievable.

FR.4 The system shall operate within the volume of the core.

Source: Project Purpose

Rationale: The system is intended to be integrated into the LIFE habitat and must fit into the dimensions of the module.

FR.5 The end-effector shall be interchangeable for modified use in future tasks.

Source: Customer Requirement

Rationale: The IVR system will be used for multiple different functions once the module is crewed including performing inspection, maintenance, repair, housekeeping, payload operations, and logistics/cargo management. The end-effector must be interchangeable so it complete different functions in the future.

FR.6 The end-effector shall be able to control and direct cargo.

Source: Project Purpose and Customer Requirements

Rationale: The end-effector needs to be able to capture, lift, and release cargo at various stages in the process. The end-effector is the only physical way to interact with cargo bags.

FR.7 The translation system shall be able to navigate from one end of the track to the other.

Source: Project Purpose

Rationale: There can be no human interaction with the system. The system must be able to translate the arm with captured cargo bag.

FR.8 The linear translator shall be able to maintain structural integrity under the torques and forces applied to it when moving cargo.

Source: Project Purpose

Rationale: Since the system will be tested in a 1G environment, it will experience more external forces than in space. The system must be physically robust enough to withstand the expected forces and torques without deformation.

3 Final Design

Authors: Lindsay Cobb, Logan Vangyia, James Tiver, Kyle Li

3.1 Requirement Flow-down

DR.1.1 The translation system shall be capable of translating the robotic arm, end effector, and cargo bag's combined mass.

Source: FR.1

Rationale: The application of this system is for zero gravity, but for test purposes all of the components must function in a 1G environment.

DR.2.1 The system shall be capable of translating a cargo bag across the track in less than 10 minutes.

Source: FR.2

Rationale: While there is no time constraint from the costumer, the system needed to operate within a reasonable time frame to be validated accurately.

DR.2.2 The cargo bag shall have a maximum weight of 1 kg.

Source: FR.2

Rationale: The system is intended for use in zero gravity where the mass of the cargo bag is not a factor, only it's inertia. Weight is not intended to be a factor for this demonstration so the bag will be as light as possible.

DR.2.3 Translation across the track stall be repeatable within 1 cm of precision.

Source: FR.2

Rationale: 1cm precision is 0.05% of the entire track length which is an acceptable amount of error since the system can recalibrate with each cycle.

DR.2.4 The length of the track shall be 1.6 m.

Source: FR.2

Rationale: The core is about 5.5m in length and the system is designed to access half of the distance. This will allow the arm to reach from the cargo hatch opening to an access way at the midpoint.

DR.3.1 The system shall identify a cargo bag with its orientation and position.

Source: FR.3

Rationale: The robotic arm needs the position and orientation of the cargo bag so it can correctly position itself for capture without colliding with the bag.

DR.3.2 The system shall be able to determine if the cargo has reached the target location

Source: FR.3

Rationale: As part of the status check, the system needs to identify if the transport task has been completed successfully.

DR.3.3 The operating system shall be able to determine if the cargo is irretrievable.

Source: FR.3

Rationale: If a cargo bag is in an orientation that cannot be grasped my the arm, all action should stop to prevent any damage to the bag or end effector. The system must also measure if the cargo bag is too far away from the arm for capture.

DR.3.4 The system shall give feedback if the transportation has failed and cease operations.

Source: FR.4

Rationale: If the bag has been dropped, or if it is irretrievable, operations need to stop so an operator can access the situation. This will prevent any possible damage to the system.

DR.4.1 A testing zone shall be defined as a cylinder with diameter 2.5m and length 3.9m.

Source: FR.4

Rationale: The testing zone will be a mock up of the core environment to complete full scale testing. This will provide the boundaries of the system so it can integrate into the core.

DR.4.2 The base of the translation system shall be less than 84 cm in width

Source: FR.4

Rationale: This is derived from other systems built into the LIFE habitat. These dimensions will prevent the system from colliding with any obstacles.

DR.5.1 The robotic arm shall be compatible with multiple end-effectors.

Source: FR.5

Rationale: The robotic arm will be re-purposed for future tasks once the habitat is crewed. It may require different end-effectors to complete other tasks.

DR.6.1 The end-effector shall secure cargo for the duration of all translation and rotation required for a task.

Source: FR.6

Rationale: The end-effector will need to keep control of a cargo bag in any orientation and while the system is translating without dropping the bag.

DR.7.1 The translator shall be able to move to different positions along the track with a bag.

Source: FR.7

Rationale: The translator will need to support the arm in what ever orientation is required for the arm to control a cargo. It should also support any additional weight.

DR.7.2 The translator shall be able to move to a prescribed location within a margin of 1 cm.

Source: FR.7

Rationale: This will account for a 0.05% error of the whole track which will not propagate an measurable error.

DR.8.1 The system shall not deflect from any torques caused by motors.

Source: FR.8

Rationale: The translator will experience torque from the track motors and the robotic arm. It must no deform from any of these external forces.

DR.8.2 The rail system shall not experience any deflection that misaligns the threads.

Source: FR.8

Rationale: The drive screw is essential to the success of the project and cannot experience any deflection that could damage the system.

3.2 Design Description

3.2.1 Overview

Once the trade studies were completed for each of the subsystems, the final design for RIVeR's project began to take shape. The major components of the design consist of a UR-10 robotic arm, a SMC MHM-32D magnetic gripper, a 1.58 meter linear actuating stage, a stepper motor, ten Pixy 2 cameras, and two Arduinos. A visual summary of this can be seen below in figure 25. This system is very software heavy which will be reviewed in detail in later sections. For now, a summary of operations consists of the Pixy 2 cameras identifying colored markers on a box that the software uses to triangulate the real three dimensional position of the box. The coordinates of the box are sent to the arm. Then the software task planner actuates the stepper motor to drive the linear platform to the location nearest the box. The pickup location is typically the same and will resemble the general location of the hatch in the core of the LIFE module. The UR-10 motion planner then maps and executes a trajectory to the magnetic plate embedded in the cargo box. Once the box is secured, the arm moves to a safe pose, and the stepper motor drives the platform to the other end of the translator. The arm then plans and executes a motion trajectory to drop of the box in the desired location. The specifics of each subsystem will be elaborated in more detail below.

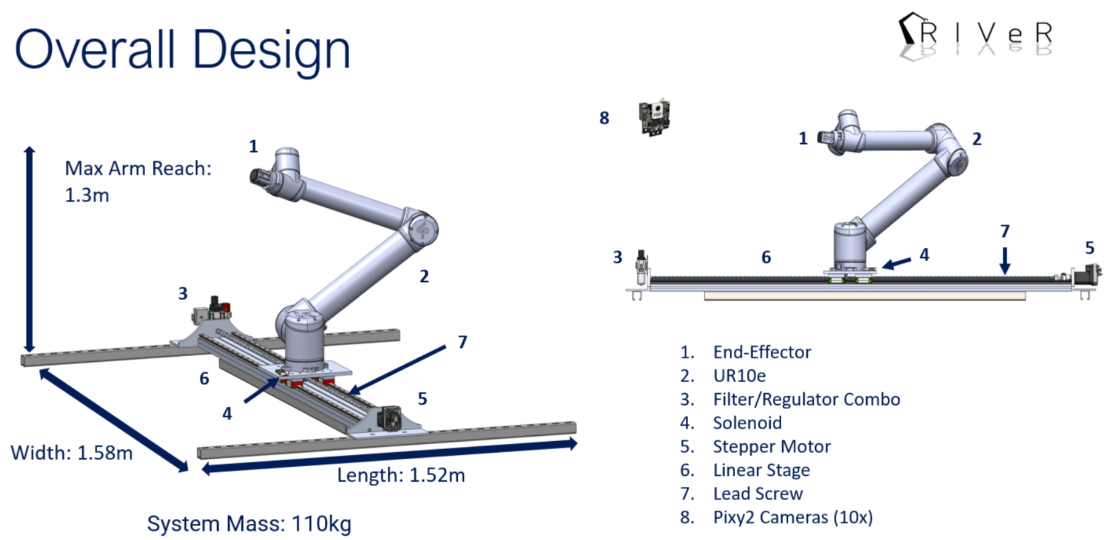


Figure 4: Final System CAD

3.2.2 Translator

The Translator subsystem consists of a number components which are labeled in figure 5 below.

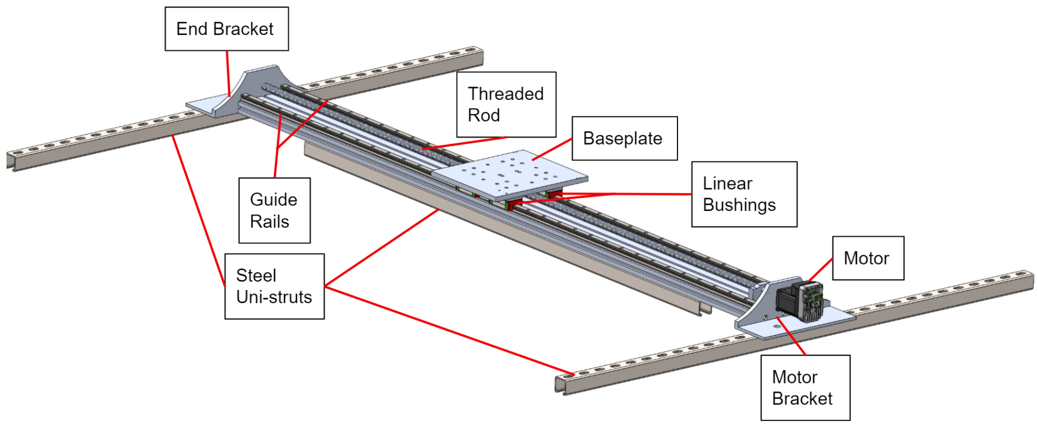


Figure 5: CAD Assembly of Translator System

The entire Translator system has a mass of about 87kg. The purpose of this system is to move the base-plate, which has the robotic arm attached to it, to any position along the length of the core. The system accomplishes this using a stepper motor that rotates a lead screw that intersects with a lead nut that is bolted to the bottom of the base-plate. This spinning rod pushes and pulls the robotic arm but is not load bearing of the downward weight of the arm. Two sets of linear bushings sliding atop a pair of guide rails to counter the moments and loads caused by the robotic arm. These forces are then directed towards the brackets on the end of the system. The original plan for this project was to have the guide rails bolted to the table or the ground that a demonstration would be given, but we are not allowed to damage anything in the senior projects room. We circumvented that issue with steel uni-strut beams that keep the system from rolling over that are attached at the ends of the system. The translator system also serves as a connection or mount for other sub-system components. The base-plate and brackets are also used to mount components necessary for the end-effector and robotic arm.

3.2.2.1 Electronics

The translator electronics are responsible for controlling and reading the stepper motor and relaying the information back to the system commanding PC.

3.2.2.1.1 Control

The translator is controlled using the Arduino Uno connected to the sensor suite. The stepper motor is a Lexium MDrive LMDCE573, Nema 23 integrated stepper motor, driver and controller. The motor also contains closed loop encoder feedback in addition to many smart features and communicates via Ethernet. In order to interface to Ethernet an Ethernet Shield V2 is attached to the Arduino. This allows the Arduino to send commands and receive encoder data from the stepper motor via UDP packets. The stepper motor and end-effector communicate with the computer via an Arduino separate from the sensor suite. Therefore, the Arduino shall be able to function as a PLC to the stepper motor and a controller to the Pixy2s without any constraint to the data budget.

3.2.2.1.2 Power

The stepper motor is powered by a dedicated 24 V power supply that is capable of up to 4A. This is plenty for the stepper motor which requires a maximum of 3A. From the motor data sheet, the power supply wire required is a braided and shielded 18 Gauge wire. This helps reduce risk of EMI which could cause motor failure. The shielding is connected to an earth ground in order to completely shield the twisted pair from outside noise.

3.2.3 End Effector

The end effector consists of three major subsystems. These are the end effector, the solenoid, and the pressure regulation system. The end effector is the magnetic head, attached to the arm tool flange that directly interacts with the cargo bags. The grab switch and actuation sensor are directly integrated on to the body of the end effector. The grab switch is activated when the cargo bag has made contact with the end effector and will stay activated until the bag is released. This allowed us to ensure that the bag as been acquired. The actuation sensor's function is to notify the system whether the magnet is in the engaged or this disengaged position. The solenoid directs air flow to control the pressure in the end effector thereby actuating and disengaging as commanded. The solenoid is controlled by a driver which sends commands from the arduino to perform the desired function by essentially powering it on and off. The pressure regulation system is divided into three components; the manual dump valve, the filter/regulator combo, and the electronic dump valve with a pressure gauge. The pressure line is sent directly from the air supply in the projects room to the manual dump valve which can be set to either "exhaust" or "supply". Exhaust vacates all air supply whereas supply allows air to proceed to the filter/regulator combo. The combo performs the function of filtering the air of any particles and moisture. Finally, the air reaches the electronic dump valve which will vacate air in case of failure. From here air continues through the solenoid and up to the end effector itself.

Figure 6: End Effector

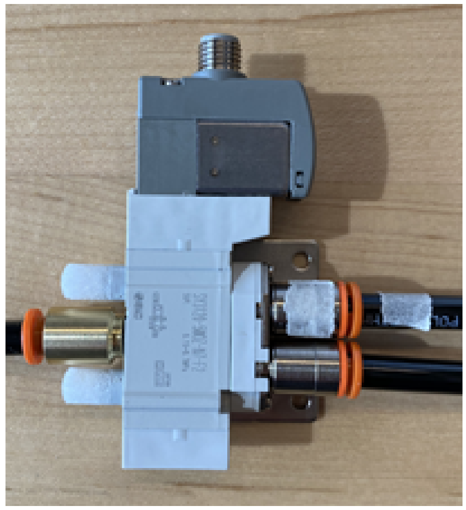


Figure 7: Solenoid (M8 connector not shown)

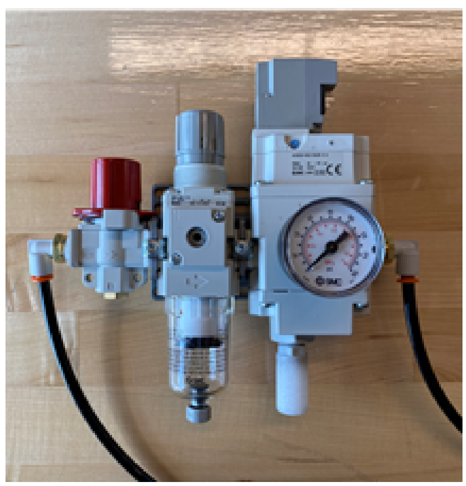


Figure 8: Left to right: Manual Dump Valve, Filter/Regulator Combo, Electronic Dump Valve

3.2.3.1 Electronics

The electronics of the end effector control and monitor the actuation of the magnet. The solenoid driver receives and sends data to and from the Arduino Uno. The driver is responsible for turning the solenoid on/off based on the received commands. The grab switch sends data to the Arduino notifying the software system on the status of the bag-magnet interface. Finally, the actuation sensor sends the status of the magnet position to the rest of the software system.

3.2.3.2 Solenoid

The solenoid acts as an electronic control valve that directs pressure to either speed controller, which in turn actuates the magnetic gripper. The solenoid chosen for our system is the SY3000 Series 5-port Solenoid Valve (SY3120-5WOZ-N7-F2). This solenoid requires 0.35 W for operation and an M8 connection for power and data transmission. It is shown in Figure **??** in the appendix.

3.2.3.3 Control

The solenoid was originally activated using a solenoid driver connected to an Arduino Uno. However, after extended use from testing, the solenoid driver eventually overheated resulting in critical failure. To compensate for this a simple switching circuit with a fly-back diode is used in the final design to switch the solenoid on and off at 24V. This circuit is activated by a digital I/O pin located on the Arduino Uno being used to control the stepper motor. The schematic for this switch can be seen in Figure 9.

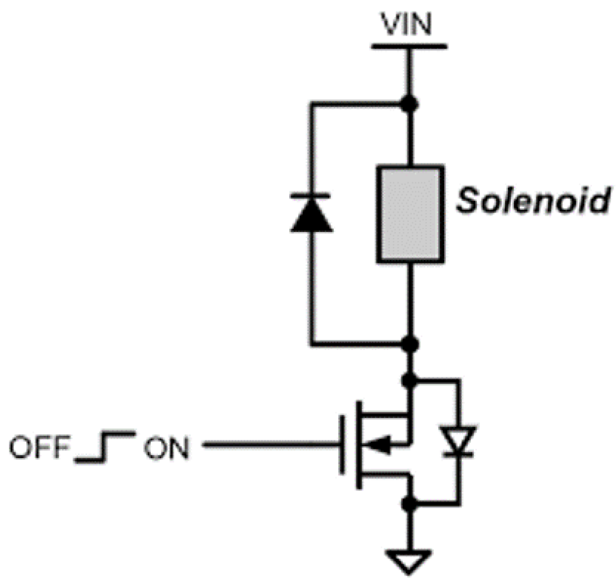


Figure 9: Solenoid driving circuit used to replace procured solenoid driver.

3.2.3.4 Power

All of the end-effector electronics components are powered by a 24 V rail coming from the UR-10 control box. The power goes directly to the auto switch, limit switch and solenoid circuit. The total power draw from the end-effector system is 20.72 W and which is easily handled by the 72 W available from the UR-10 control box.

3.2.4 Sensors

The sensors utilized consisted of ten Pixy 2 cameras 11a, three limit switches 12a, and an actuation sensor 12b. Five of the Pixy 2 cameras were located around one half of the core to locate a box in the pick up area. The other five cameras were located in the other half of the core to verify when the cargo made it to the drop off area. The camera positions can be visualized in below in figure 10

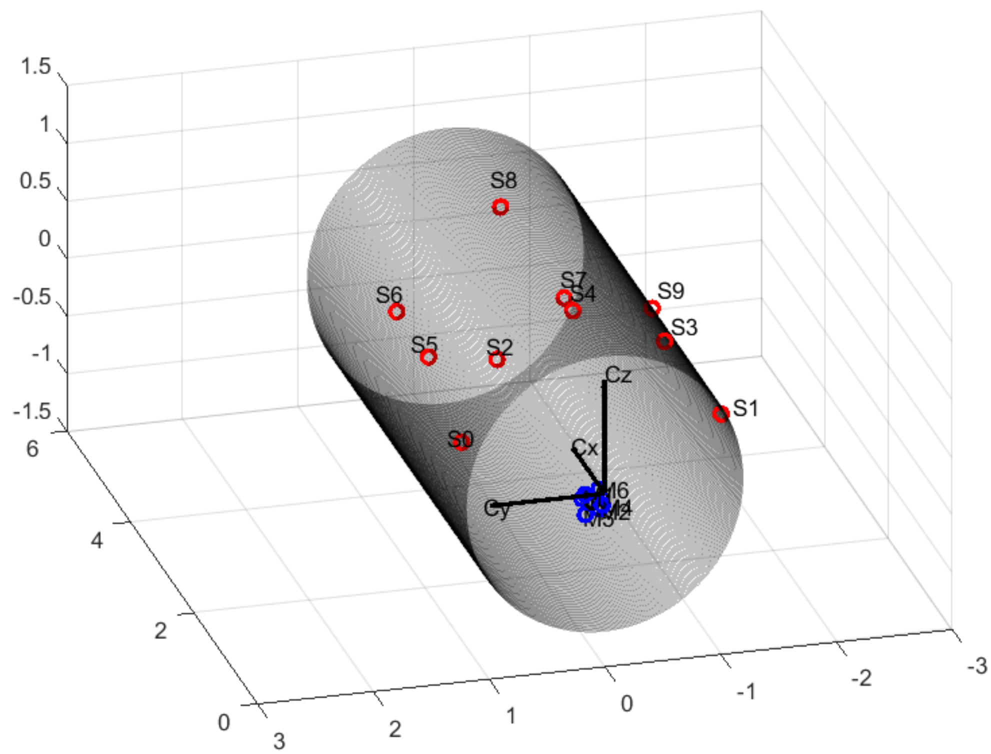
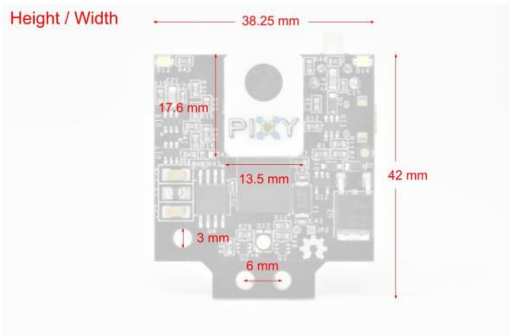
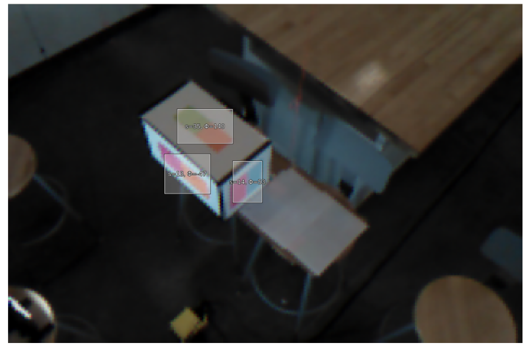


Figure 10: Sensor Positions

The cameras were wired via the I2C lines to the Arduino. All of the camera wires were split and connected into the same I2C ports of an Arduino Mega. The cameras were chosen for their capability of recognizing colored markers. An example of the color recognition can be seen in figure 11b. The Pixy outputs X,Y pixel coordinates of each visible color coded marker. The software algorithms use this to triangulate the markers to locate the box in three dimensional space. More of the triangulation will be discussed in the software manufacturing section.

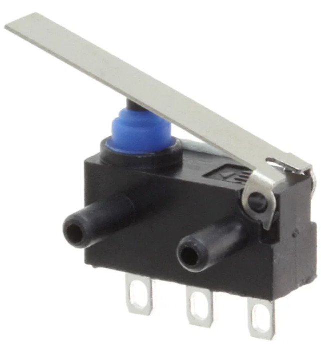


(a) Pixy 2 Schematic

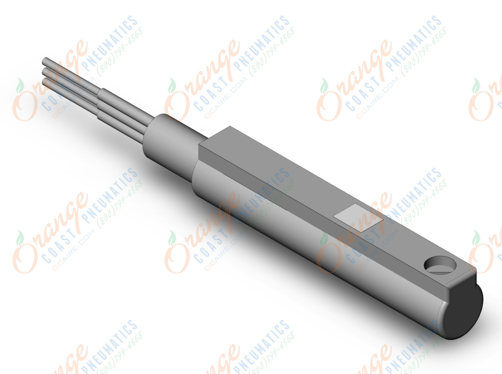


(b) Pixy 2 Color Recognition

Next, the limit switches were used as push sensors for the translator and the end effector. Two of the limit switches were placed on either end of the translator to prevent the robotic arm on the platform from colliding into the ends of the tracks. When the limit switch is pressed a digital high signal is sent to the connected device confirming the switch is activated. The two translator limit switches are connected to the stepper motor and when pressed, automatically turn off the motor to prevent any damage to the system. The third limit switch is used as a push sensor attached to the end effector. This switch is intended to tell the system when a box is picked up. When the magnetic end effector actuates and grabs the box, it will push against the sensor. A digital high signal will be sent to the Arduino which will forward the signal to ROS to tell the system whether a box is being held by the robotic arm or not. Finally the actuation sensor is also attached to the end effector. The actuation sensor sends a digital high signal to ROS via the serial Arduino connection to notify the system when the magnet on the end effector is actuated or not.



(a) Limit Switch



(b) Actuation Sensor (SMC D-M9PL)

3.2.4.1 Electronics

The sensor suite electronics are tasked with providing the interfaces necessary to control and power 10 Pixy2 Cameras.

3.2.4.1.1 Control

The cameras communicate with an Arduino MEGA using an I2C bus. The I^2C bus which consists of an SCL and SDA signal. The SCL signal functions as the clock and the SDA is a data signal. Both of these signals are sent through a 24 gauge wire that is split using T shape connectors and twisted wire connectors. Each device on an I^2C bus contains a unique address that the controller can identify. The Pixy2 can be configured up to 127 addresses which is excessive for the 10 cameras in our system. Each Pixy2 contains a $87k\Omega$ pull-up resistor for the SDA and SCL signals so that an additional circuit is not required. Each Pixy2 is capable of communicating at up to 115 kbps. Therefore, in order for the Arduino to process the Pixy2 data in series it must be capable of 1.15 Mbps. The Arduino MEGA used is capable of 2 Mbps providing a .85 Mbps margin for other processes. Once I^2C data is recieved by the Arduino it processed and sent to the computer via USB serial.

3.2.4.1.2 Power

The Pixy2 cameras can be powered by USB, a regulated 5V line, or an unregulated 6-10V line. For simplicity and reliability USB power is utilized on this project. Each Pixy2 camera shall has a dedicated micro USB cable connected to a USB hub connected to the same laptop used to command the Arduino. It is imperative that the Pixy2 Camera's are powered from the same source as the Arduino MEGA in order for the I2C line to have a shared reference ground. Each Pixy2 camera requires .14A at 5 VDC and the adapter selected is capable of providing 5 VDC with 2.4 A per port. Each Pixy2 camera kit comes with a 1 meter micro USB capable. For the five cameras furthest the stepper motor the team procured additional 15 foot micro USB cables.

3.2.5 Software

The decided software utilized in this project was Robotic Operating System (ROS). The version chosen was ROS melodic and it was installed on an Ubuntu 18.04 operating system ran on a laptop. ROS was chosen for it's extended history within robotic systems. ROS allows for multiple pieces of hardware and software to autonomously operate and communicate together. Then to incorporate the Arduino communications into the entire system ROS-serial was utilized. This allowed sensors connected to the Arduino to communicate with the rest of ROS through the USB cable via a serial protocol. The major sub components of the software include an Arduino node, a task planner node, a bag configuration node, a manipulator node, and a status node. Below a software flowchart visualizes an overview of the software communication in Figure 13. The specifics of the nodes and their development will be further elaborated below in the manufacturing section.

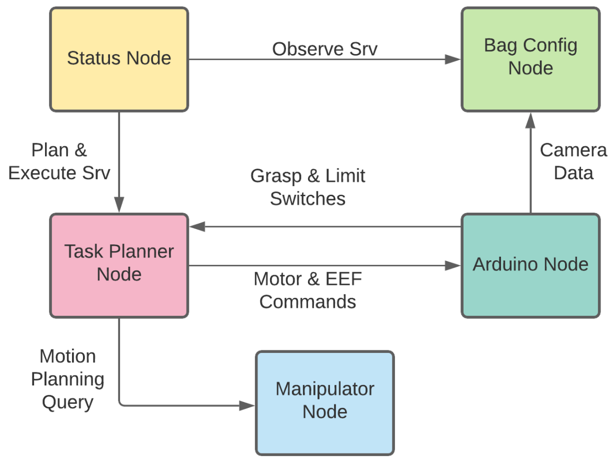


Figure 13: ROS Communications

3.2.6 System Electronics

3.2.6.1 Control

The primary control of the system is a laptop that will run our ROS software. This PC uses an Ethernet signal to send and recieve packages from the UR-10. Additionally, there is an Arduino Uno used to control the End-Effector and Stepper Motor that communicates with the PC via USB and an Arduino Mega used to interface the Pixy 2s to the PC via USB. The signals received by the Arduino Uno are Ethernet UDP packets and digital logic from the limit switches and End Effector. The signal received by the Arduino Mega is an I2C bus containing data from all 10 Pixy2 cameras. The pinout for the stepper motor Arduino can be seen on Figure 14

Pin	Function	Device
Digital 5	Logic High/Low(Output)	Solenoid Driver
Digital 7	Logic High/Low(Input)	Auto Switch
Digital 8	Logic High/Low(Input)	Positive LImit Switch
Digital 9	Logic High/Low(Input)	Negative Limit Switch
Digital 10	SS	Ethernet Shield
Digital 11	MOSI	Ethernet Shield
Digital 12	MISO	Ethernet Shield
Digital 13	SCK	Ethernet Shield
Ethernet Port	UDP	Stepper Motor

Figure 14: Arduino Uno Pinout responsible for processing and commanding data from the stepper motor, limit switches, activation switch and solenoid driver.

3.2.6.2 Power

There are three primary power sources for the system: the UR-10 control box, the stepper motor power supply and USB hub. The UR-10 control box powers the UR-10,limit switch, LEDs and end-effector components. The stepper motor power supply is dedicated to just the motor which requires 3A at 12-48V. The USB power source powers the 10 Pixy2 cameras and is connected to the PC USB port in order to provide a common ground. The UR-10 control box powers the UR-10 and end-effector at 24V without any additional converters. In order to power the LEDs, the UR-10 24V line must be passed through 2 Boost converters that increase the voltage to 30V.

3.2.6.3 Wiring

In the final system there are many types of wire used. There is 250 ft of 24 gauge wire used to power the limit switches and transmit I2C data lines. Additionally, there is 50 ft of 14 gauge wire to power the LEDs mounted on the sensor cage. To power the stepper motor, an 18 gauge braided and shielded wire is used to reduce noise. To communicate with the stepper motor a straight-through Ethernet cable is used instead of a crossover Ethernet cable. When initially using a crossover Ethernet cable there were problems that occurred due to neither the Ethernet Shield or stepper motor containing auto MDI-X capability. Therefore, it is imperative that a straight-through cable be used for this application. To split the 24 gauge wires T2 connectors were initially used. These connectors were difficult to use and created many unexpected shorts. Due to these issues most of the T2 connectors were replaced with a standardized twist on wire connector in the final design.

3.3 Trade Studies

All relevant trade studies are included in the Appendix section A.

4 Manufacturing

Authors: Jordan Abell, Jett Moore, Logan Vangyia, James Tiver, Brandon Torczynski

4.1 Hardware

Figure 15 shows components of the project that were purchased. Given the complexity of design and the uncertainty around COVID-19, the team decided to purchase and integrate many components rather than designing and fabricating them directly. This was mostly due to time constraints and access to the aerospace machine shop at CU.

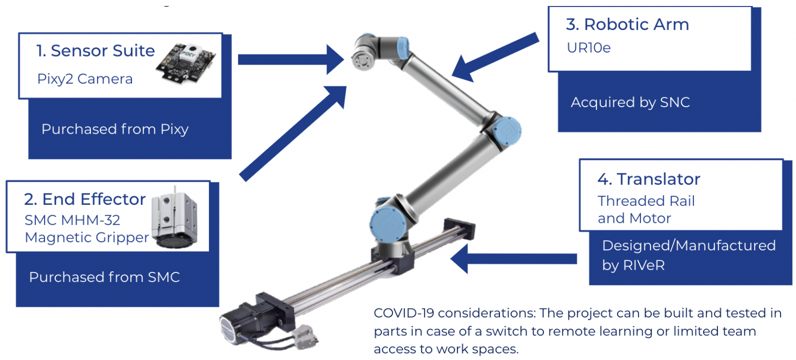


Figure 15: Purchased Components

4.1.1 Manufactured

The blue components of figure 16 represent portions of the translator that were fabricated out of aluminum and are described in greater detail in the mechanical section.

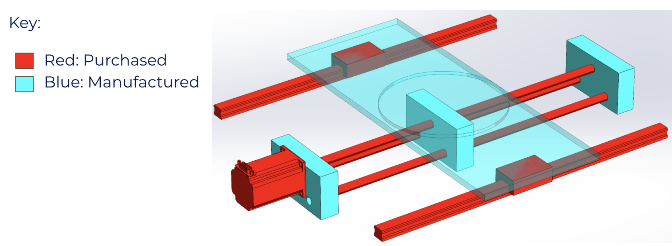


Figure 16: Manufactured Parts

4.1.1.1 Mechanical

The mechanical components that we designed and manufactured were the baseplate (figure 17), motor bracket (figure 18), and end bracket (figure 19).

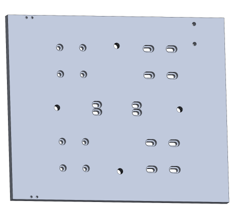


Figure 17: Baseplate CAD

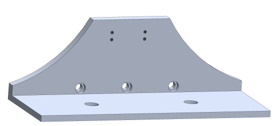


Figure 18: End Bracket CAD

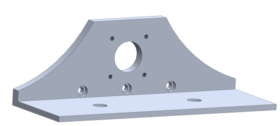


Figure 19: Motor Bracket CAD

All parts were made from half inch aluminium 6061 that were submitted to the machine shop for fabrication. The baseplate's purpose was to replace the piece of metal that came stock on the translation system we ordered. Our baseplate was larger and had built in threads to mount the robotic arm as well as a few other small components that needed to be close to the arm. Both brackets bolted to the ends of the translator's track in order to hold the system together. The motor bracket was also responsible for holding the motor steady while in use. The end bracket was responsible for mounting the dump valve for the end effectors pressure regulation. The challenges faced during this design were related to tolerances in design measurements when describing the design to the machine shop so that they understood what we wanted out of the design.

4.1.2 Purchased Translator Parts

There were two main components that the translator sub-team purchased. The first was the main stage of the translator that included the rails, bushing, lead nut, and threaded rod. We chose to purchase these components together in order to simplify the integration of the entire system since finding parts that were compatible was impossible when looking at them online. Also, we did not have the particular skill set to accurately design this type of system from the ground up. The other major component purchased for the translator system was our NEMA 23 stepper motor.

4.2 Electrical

The electrical subsystem is primarily concerned with the interfacing of components signals and power. For the sensor cage, wire connections were made using T2 connectors and twist on connectors. Twist on connectors worked more effectively between these two options. Before going to the Arduino signal buses were soldered to a protoboard that was used for limit switches and the solenoid circuit. Initially the solenoid driver was contained on a procured PCB, but after failure the circuit shown on Figure 9 was assembled on the protoboard. This circuit consists of a transistor with the base connected to the Arduino and the emitter outputting 24V to the solenoid. Additionally there is a fly-back diode across the solenoid in order to prevent an inductive load.

4.2.1 Purchased Electrical Components

Table 2: Procured Electrical Components With Part Number

Item	Part Number	**Supplier**	**Description**
Stepper Motor	LMDCE573	MSITec	Lexium MDrive Nema 23 Stepper Motor that uses Ethernet Protocol
Limit Switch	WS0850101F050SA	Digikey	Limit switch used for translator location and end effector grasp confirmation.
Arduino Uno R3	A00024	Arduino	Arduino Uno Used to command the translator and end effector subsystems.
Arduino Ethernet Shield	A000024	Arduino	Ethernet Shield used to interface the Arduino to the stepper motor
LED 100W Chip	LH-XP-100W-6000k	Amazon	LED lights mounted on the sensor cage in order to improve Pixy Accuracy.
Voltage Booster	B08246FB95	Amazon	Voltage booster used to convert from the 24V supplied by the UR-10 to 30V required by the LED Chips
AC/DC Converter	ALT-2405	Amazon	AC/DC Converter used to supply power to the stepper motor.

4.3 Software

4.3.1 Estimation Algorithms

One of the core functionalities of the RIVeR project is the ability for the sensor suite to identify and estimate both the position and orientation of cargo bags in the LIFE module core. This capability is achieved using two estimation algorithms, one to estimate bag marker positions, and the other to estimate the bag orientation.

4.3.1.1 Position Estimation

This estimation problem consists of a static target (the cargo bag) with defined markers, and static sensors (the Pixy2 cameras) scattered around the Core. This problem is nonlinear due to the geometry of the objects and the visibility cones of the sensors. Therefore, a nonlinear estimation method is required to locate a cargo bag. An Extended Kalman Filter (EKF) was designed to solve this estimation problem to estimate the 3D positions of visible markers on the cargo bag.

4.3.1.1.1 Nonlinear Measurement Function

For brevity, only a summary of the formulation of this estimation problem is provided in this paper. Each Pixy2 camera provides a 2-dimensional measurement of identified markers in a camera-body coordinate frame. A vector pointing from the *ith* camera to the identified marker can be written as:

$$\mathbf{v}_{si} = a \begin{bmatrix} tan(\alpha) \\ tan(\beta) \\ 1 \end{bmatrix} = a \begin{bmatrix} 2tan(\gamma)(\frac{x_{si}(k)}{\Delta x} - \frac{1}{2}) \\ 2tan(\theta)(\frac{y_{si}(k)}{\Delta y} - \frac{1}{2}) \\ 1 \end{bmatrix} \tag{1}$$

where a is a normalizing constant to ensure the vector magnitudes are the same, α is the angle between the vector $\mathbf{v}_{si}$ and the vector $\hat{\mathbf{z}}_{si}$ in the x-z plane, γ is the horizontal half field of view angle, $x_{si}(k)$ is the kth measured x value, Δx is the maximum x measurement, β is the angle between the vector $\mathbf{v}_{si}$ and the vector $\hat{\mathbf{z}}_{si}$ in the y-z plane, θ is the vertical half field of view angle, $y_{si}(k)$ is the kth measured y value, and Δy is the maximum y measurement.

Figure 20 shows a visual representation of the problem geometry.

The vector $\mathbf{v}_{si}$ can equivalently be written:

$$\mathbf{v}_{si} = S_i^T C^T (\mathbf{x} - \mathbf{s}_i) \tag{2}$$

where S is the coordinate frame matrix for the ith sensor ($S_i = [\hat{\mathbf{x}}_{si}, \hat{\mathbf{y}}_{si}, \hat{\mathbf{z}}_{si}]$), C represents the coordinate frame matrix for the global Core Frame (defined to be the identity matrix), $\mathbf{x}$ is the position vector of the marker represented in the Core Frame, and $\mathbf{s}_i$ is the position vector of the ith sensor represented in the Core Frame.

We can define the nonlinear measurement function $\mathcal{H}_{si}(\mathbf{x})$ by equating Equations 1 and 2 and solving for $\hat{\mathbf{y}}_{si}(k) = [x_{si}(k), y_{si}(k)]^T$ in terms of the marker position vector $\mathbf{x}$ and the problem geometry. In order to equate the two vector expressions, we must set the normalizing constant to the 3rd index of Equation 2, $a = \hat{\mathbf{z}}_{si} C^T (\mathbf{x} - \mathbf{s}_i)$. The final expression for the nonlinear measurement model is given by equation 3.

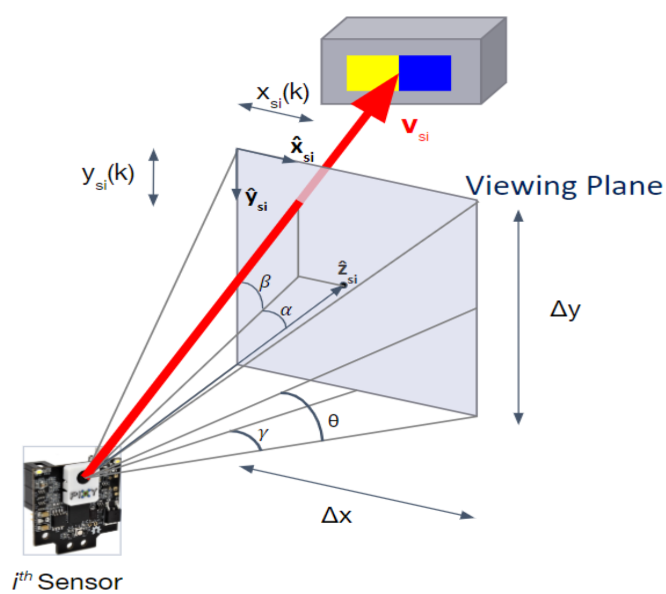


Figure 20: Estimation Problem Geometry

$$\hat{\mathbf{y}}_{si}(k) = \mathcal{H}_{si}(\mathbf{x}) = (\Delta^T \Delta)^{-1} \Delta^T \left(\frac{1}{a} T^{-1} S_i^T C^T (\hat{\mathbf{x}} - \mathbf{s}_i) + \vec{i} \right) \tag{3}$$

$$\Delta = \begin{bmatrix} \frac{2}{\Delta x} & 0 \\ 0 & \frac{2}{\Delta y} \\ 0 & 0 \end{bmatrix} \qquad T = \begin{bmatrix} tan(\gamma) & 0 & 0 \\ 0 & tan(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \vec{i} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

4.3.1.1.2 Estimation Filter

The nonlinear measurement function $\mathcal{H}_{si}(\mathbf{x})$ in Equation 3 can now be used in an Extended Kalman Filter (EKF) [3] to get an estimate of the 3-dimensional position of the measured marker. The EKF algorithm is given by the following steps:

Step 1: Initialization

Define the initial state estimate and initial covariance matrix.

$$\hat{\mathbf{x}}_0^+ = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad P_0^+ = \begin{bmatrix} \delta x_0^2 & 0 & 0 \\ 0 & \delta y^2 & 0 \\ 0 & 0 & \delta z^2 \end{bmatrix}$$

In our application, the initial uncertainties $\delta x_0, \delta y_0$, and δz_0 are determined by the physical constraints of the Core environment, so we set $\delta x_0 = 5$ m and $\delta y_0 = \delta z_0 = 1.195$ m.

Step 2: Dynamics Prediction Step

Our system is completely static during the estimation phase of the system, so there are no dynamics to predict.

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k^+ \qquad P_{k+1}^- = P_k^+$$

Step 3: Measurement Update

This estimation problem for a single marker is only observable if at least two sensors provide measurements of that marker, meaning that the Measurement Update step can only be executed when two sensors return measurements of a common marker. For each marker that has measurements from at least two sensors, the Measurement Update is executed:

$$\mathbf{y}_{k+1} = \begin{bmatrix} x_{s0}(k+1) \\ y_{s0}(k+1) \\ \vdots \\ x_{sn}(k+1) \\ y_{sn}(k+1) \end{bmatrix} \qquad \hat{\mathbf{y}}_{k+1}^- = \mathcal{H}(\hat{\mathbf{x}}_{k+1}^+)$$

$$\tilde{H}_{k+1} = \left.\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k+1}^-} \qquad \tilde{K}_{k+1} = P_{k+1}^- \tilde{H}_{k+1}^T [\tilde{H}_{k+1} P_{k+1}^- \tilde{H}_{k+1}^T + R_{k+1}]^{-1}$$

$$\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \tilde{K}_{k+1}(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1}) \qquad P_{k+1}^+ = (I - \tilde{K}_{k+1}\tilde{H}_{k+1})P_{k+1}^-$$

The measurement update step is repeated for each set of measurements returned by the sensor network until the variance bounds are small enough to satisfy accuracy requirements.

4.3.1.1.3 Calibration

The EKF algorithm used in the above section requires an accurate knowledge of each sensor's position and pointing to return useful results. However, setting up the sensor network at distances multiple meters from the reference origin (the Core's origin) is a difficult and imprecise task in 3D space. This issue leads to the necessity of sensor calibration before marker position estimation can be done. The marker estimation algorithm uses the known positions (and orientations) of each sensor and sensor data to estimate the position of a marker. Conversely, we can use the known position of markers and sensor data to estimate the positions of each sensor. The process is very similar the the estimation algorithm described above, with slight differences. Now, instead of our nonlinear measurement function $\mathcal{H}$ being a function of the marker position, we write it as a function of the sensor position, $\mathcal{H}(\mathbf{s})$. Now, the calibration Measurement Update step becomes:

$$\mathbf{y}_{k+1} = \begin{bmatrix} x_{s0}(k+1) \\ y_{s0}(k+1) \\ \vdots \\ x_{sn}(k+1) \\ y_{sn}(k+1) \end{bmatrix} \quad (17) \qquad \hat{\mathbf{y}}^-_{k+1} = \mathcal{H}(\hat{\mathbf{s}}^+_{i,k+1})$$

$$\tilde{H}_{k+1} = \left.\frac{\partial \mathcal{H}}{\partial \mathbf{s}}\right|_{\hat{\mathbf{s}}^-_{i,k+1}} \qquad \tilde{K}_{k+1} = P^-_{k+1}\tilde{H}^T_{k+1}[\tilde{H}_{k+1}P^-_{k+1}\tilde{H}^T_{k+1} + R_{k+1}]^{-1}$$

$$\hat{\mathbf{s}}^+_{i,k+1} = \hat{\mathbf{s}}^-_{i,k+1} + \tilde{K}_{k+1}(\mathbf{y}_{k+1} - \hat{\mathbf{y}}_{k+1}) \qquad P^+_{k+1} = (I - \tilde{K}_{k+1}\tilde{H}_{k+1})P^-_{k+1}$$

We can use a single object with multiple markers (at known locations) to calibrate the network of sensors. For the calibration process to work, each sensor must be pointed at a defined point known, defined in the Core Coordinate Frame, and the top edge of each sensor must be level to the ground. These two parameters define the pointing frame of each sensor, which allows us to calibrate the sensor's positions.

4.3.1.2 Orientation Estimation

The Orientation Estimation algorithm uses the output of the Position Estimation software to determine the orientation of the cargo bag. The algorithm uses the known definition of the cargo bag, defining the relative positions between each bag marker.

The input to the Orientation Estimation algorithm is the output of the Position Estimation software. For the Orientation Estimation to return a useful result, it must be provided the 3D positions and marker IDs of three separate bag markers.

The Orientation Estimation algorithm uses a gradient descent approach to minimize the euclidean distance error between the bag definition and the marker positions output by the Position Estimation.

The gradient step is computed using the average of the cross products between the estimated and the defined position vectors of each bag marker. A user-defined number of gradient steps are executed, then the converged bag orientation is returned. The Orientation Estimation algorithm outputs a unit quaternion representing the bag's orientation.

4.3.2 ROS

The ROS environment consists of a layout of multiple custom ROS nodes that exchange information and commands in real time. The ROS nodes developed by the team are: Task Planner Node, Status Node, Manipulator Node, Bag Configuration Node, and Arduino Node.

4.3.2.1 Task Planner Node

The Task Planner Node utilizes a developed C++ Task Planning Library (external to ROS) to autonomously plan a sequence of discrete action primitives that the system must execute to complete the task. The Task Planner Node is responsible for reading the cargo location information packaged by the Bag Config Node, and creating the discrete planning environment. At this point the functionality of the Task Planning Library is used to obtain a discrete action primitive sequence. The Task Planner Node then enters the "task execution" phase. For each action within the action sequence plan, the task planner sends a command to the respective node, along with other necessary information. In order to execute the action "grasp" (grabbing the cargo), the Task Planner Node sends a command to the Arduino Node to engage the end effector, then waits for a response from the grasp switch. In order to execute the action "release" (disengage the end effector), the Task Planner Node sends a command to the Arduino Node to disengage the end effector, then waits for the reading on the grasp switch to verify that the cargo has indeed been dropped. In order to execute the action "move" (move the arm to a specific pose) the task planner must send goal configuration information to the Manipulator Node as well as planning environment information, such as whether or not the arm is in the drop-off or pickup domain. In order to execute the action "translate" (translate the arm to a certain location along the linear stage), the Task Planner Node sends a command to the Arduino Node to start the translation process, then waits for a response from the corresponding limit switch sensor along the translator, verifying that the arm has finished translating.

4.3.2.1.1 Task Planning Library

The Task Planning Library used for this cargo transfer system was developed during the spring semester along with development of the ROS architecture. It should be noted that the Task Planning Library is a C++ library that is external to ROS, and is modular and extensible to any autonomous discrete action planning problem. The application of the Task Planning Library to the ROS architecture used specific problem domain definition statements that are specific to a robotic manipulator using an end effector on a translator, translating between two locations of interest (in this case the "pickup" and "drop off" areas). In order to plan a discrete action sequence, the user must define a discrete state space for planning. Then a set of valid pre- and post- conditions must be defined for valid state transitions. The Task Planning Library includes a custom logic parser that uses reasoning groups and tools for variables and dimensions in the state space. The pre- and post- conditions are defined in terms of this logic parsing tool. A task specification must also be defined in the form of Deterministic Finite Automaton (DFA). The task specification used for our task was defined as a Linear Temporal Logic formula then converted to a DFA using *Spot's Online LTL Toolset* (see appendix). The task of "all cargo must eventually end up in the drop off area" was defined as $F(p)$ where p is the atomic proposition: "all cargo is in the drop off domain". This proposition was defined using the same custom logic parser for the pre- and post-conditions. After defining the specification, the user can plan a discrete task sequence that adheres to the physical state transitions defined by the pre- and post-conditions and the task specification.

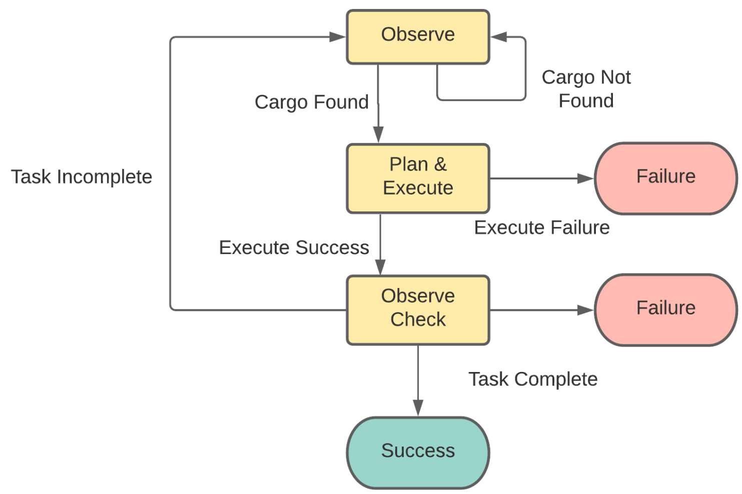


Figure 21: Status Node Operations Graph

4.3.2.2 Status Node

The Status Node is responsible for maintaining the overall status of the system, and the sequence of operations. Figure 21 displays a graph that dictates the operations of the entire system. All of the yellow boxes represent operations, then the edges represent the result of that operation. The operation "Observe" corresponds to collecting data then processing the data through the Bag Config Node until the cargo has been successfully observed. The operation "Plan Execute" corresponds to initiating the Task Planner Node, which in turn plans an action sequence, then executes the action sequence. The operation "Observe Check" corresponds to observing the environment to check that the task has been completed. This operation was not implemented in our projected due to time constraints. The Status Node handles failure of any of the operations, such that an overseeing user can determine the status of the system. The overseeing user does not need to give any input other than initiating the software.

4.3.2.3 Manipulator Node

The Manipulator Node is responsible for commanding the robotic arm. The Task Planner Node will send the manipulator node a "planning query" that contains enough information for the motion planner to plan for a path to the desired end effector configuration.

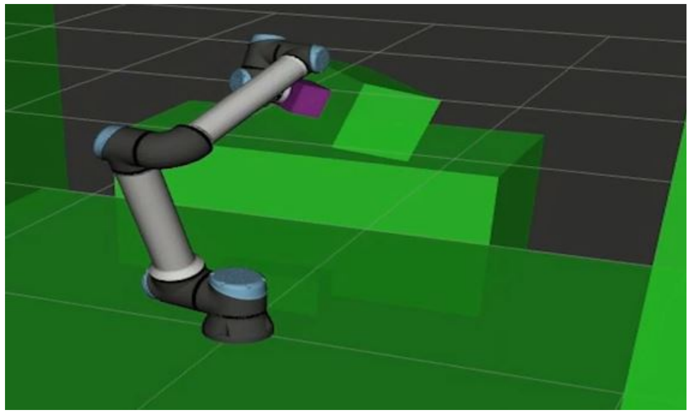


Figure 22: Motion Plan Execution Displayed in RViz

Figure 22 displays a motion plan execution in a graphical visualization tool (RViz). The Manipulator Node must define the collision environment for the corresponding location domain ("pickup" or "drop off"). All collision obstacles are defined within the node and added to the corresponding location domain. For each planning query, the Manipulator Node will load the corresponding collision environment based on the location domain sent within the planning query. The manipulator node will also define the cargo based on the observed configuration. For actions "grasp" and "release", the cargo bag is either attached or detached to the planning group for the arm respectively. For the action "move", the arm will move to the desired pose or end effector configuration. The manipulator node will return failure if a motion plan is not found within a user-defined upper bound for number of planning attempts. Generally, for simple motion planning problems, if the user gives the motion planner a large amount of time to plan, failure means that no motion plan exists. In that case, the steel plate on the bag is said to be "unreachable", and thus returning failure is accurate.

4.3.2.4 Bag Configuration Node

The Bag Configuration Node is responsible for estimating the position and orientation of the cargo bag. The Bag Configuration node receives the sensor data from the Arduino Node to develop the state estimate of the cargo bag. It sends its final state estimate of the cargo bag, including the position of the bag center and the bag's orientation, to the Task Planner Node.

The Bag Configuration Node begins estimation when it receives a service request from the Status Node, indicating that the system is in an 'Observe' state. Once the Bag Configuration node has finished observing the core and has a state estimate of the bag, it publishes a message to a specified ROS topic for the Task Planner node to use.

The Bag Configuration node is an envelop node to run the Position Estimation and Orientation Estimation algorithms, as described in Section 4.3.1.

4.3.2.5 Arduino Node

The Arduino node is responsible for handling the communications between the Arduino and the computer ROS interface. This handles the communications for the sensors linked to both Arduinos. The communications are primarily handled by a ROS package known as ROS-serial. This package allows for most of the ROS functionality to be imported onto the Arduinos to be coded up in a similar method as ROS C++. The primary sensors are the Pixy 2 cameras which are wired to one Arduino. Then the stepper motor, limit switches, and actuation sensor are wired to the second Arduino. On the Arduino with the Pixy cameras the ROS communications primarily utilized a publisher to one topic to push the Pixy 2 data to the computer. The ROS message type utilized was an int64 array message that sent the data in the form of a 1x4 vector in the form of x, y, color marker ID, sensor ID. Originally the intended message was to send a multi array message, but unfortunately the Arduino was incapable of sending two dimensional arrays of variable length. The ROS-serial client could only send 2-D arrays parsed out into a 1-D array of fixed length. This was avoided because of potential difficulties properly unpacking the data leading to mismatched data for the estimation algorithm. The second option was to organize the data using custom ROS messages by embedding an array message inside an array message. However, ROS serial had several issues with the compatible ROS melodic version. The issues included not being able to properly send custom messages nor services and clients. As a result, the Arduino node encompassed a C++ script on the computer side that subscribed to the ROS topic that the Pixy data was published to. Then the script would repackage the data to group data points from the same sensor together. Then the groupings of data from each sensor would be packaged together in another ROS message and would all be published to another topic that the bag config node would subscribe to.

Next, the Arduino node handles the communications between the stepper motor, the limit switch, the actuation sensor, and the computer ROS environment. The limit switch and actuation sensor communicate via boolean signals with the Arduino. To communicate the sensor data to ROS, a 3x1 array message was utilized. The first entry was an integer value that represented a preset position command for the stepper motor. The Arduino code and the task planner maps the integer value to a specific position value along the translator. The second entry is a boolean value for a limit switch attached to the end of the end effector. Then the third entry is a boolean value for the actuation sensor that tells if the end effector is actuated or not. The combination of the two boolean values is used to determine if the end effector has successfully grasped onto the cargo bag or not. This message of sensor data is published to a ROS topic that the status node would subscribe to. Then to operate the end effector or stepper motor a separate message is published from the task planner that the Arduino node subscribes to. This message is also a 3x1 array where the first entry is a boolean value where 0 tells the system to operate the end effector and 1 to operate the stepper motor. The second entry is also a boolean value that tells the system to engage or disengage the end effector. The third entry is a 64 bit integer value that tells the stepper motor where to drive the linear stage to. The integer value is mapped to a hexadecimal position command on the arduino that is then forwarded through a UDP command to the stepper motor. Since the system only needs a few preset positions, the software was able to just have preset conditions as opposed to dynamically setting the translator position.

4.4 Testing Environment

In an effort to best mimic the actual environment, a crude mock-up of the core of the LIFE module was crafted utilizing two by fours and PVC piping. The purpose of the mach up was to mimic the circular nature of the cylindrical core. The PVC piping mounted to the the two by four frame outlines the circular diameter of the core of the life module. The Pixy 2 cameras were mounted to the PVC piping by attaching two L brackets together and screwing them to the camera and to the PVC pipe. Due to the simplicity of the mach up core, no CAD models were developed for the structure. To show the similarity of the LIFE modules core and the mach up please refer to figure 23 below.

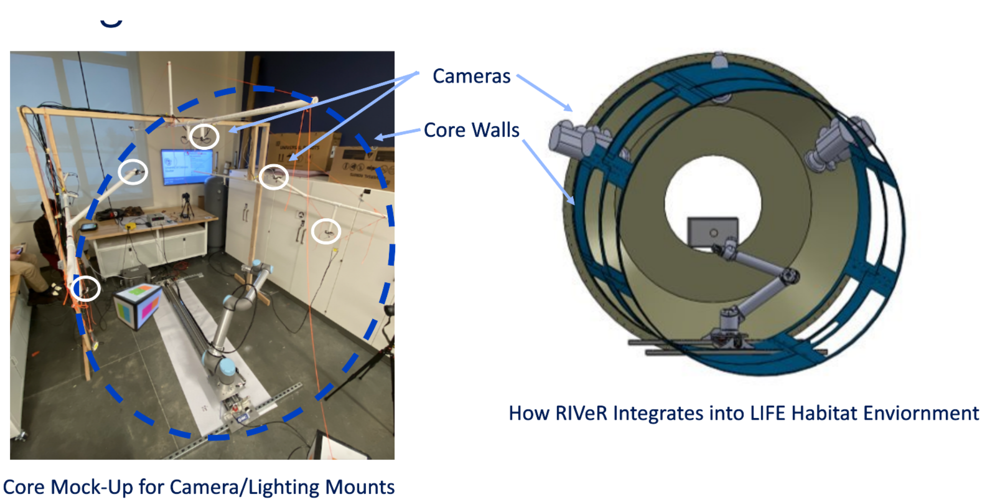
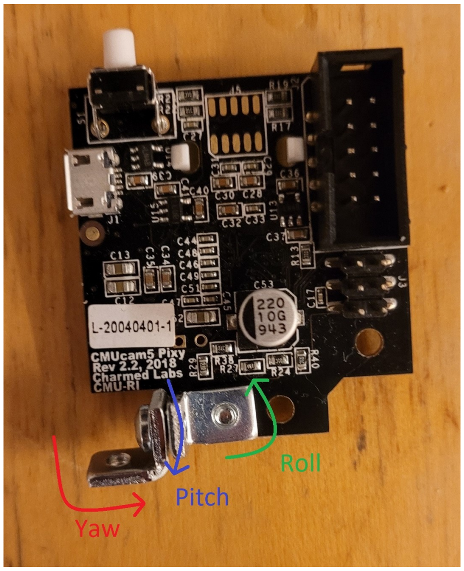
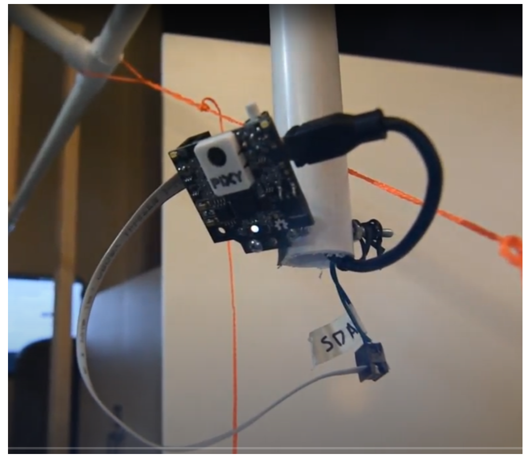


Figure 23: Testing Environment Comparison

The testing environment helped facilitate the adjustments of the cameras. The PVC legs that the cameras were attached to could be rotated in conjunction with the L brackets of the cameras to allow rotations of the cameras in all three dimensions. The first dimension adjustment is done by rotating the camera along the screw that attaches the camera to the L bracket to give the Pixy a roll adjustment. The second dimension is the rotation provided by twisting the L bracket about the 2nd L bracket which pitches the camera forward or backwards. Then the third is done by rotating the L bracket that screws into the PVC piping which provides a yaw rotation. Refer to figure 24a below for a visualization of the mounting solution and roll, pitch, yaw adjustments. Then refer to figure 24b to see the camera mounted to the testing environment.



(a) Pixy 2 Mounting Roll, Pitch, Yaw



(b) Pixy Mounted to Testing Environment

This simplistic mounting solution was utilized due to a limited budget. If more money was available some sort of 360 degree swivel mount similar to those used on camera tripods would have been picked. This would allow easier articulation without having to loosen screws to adjust the camera or the L brakcets then to tighten them again. Occasionally the screws would get stripped out and have to be replaced due to constant adjustment. Despite the difficulty and less optimal selection of the mounting solution, it was still sufficient enough to serve its purpose.

4.5 Integration

By integrating all these components our final design allows for translation, sensing, and grabbing. The translator gives the arm the ability to translate along an axis, the cameras and ROS in conjunction allow for cargo to be detected and for obstacles to be avoided. The end effector enables the arm to grab cargo, ultimately resulting in the design:

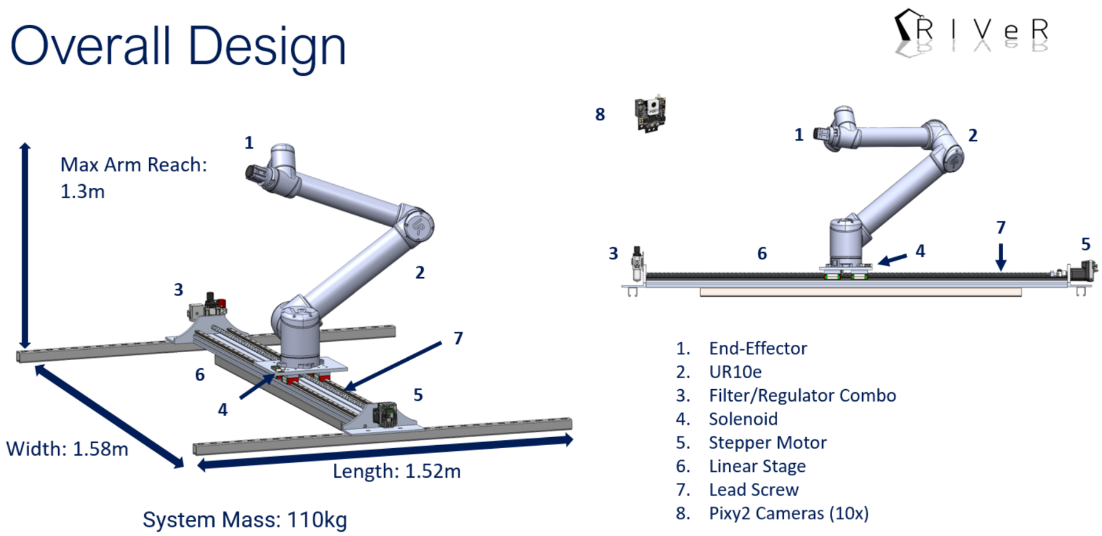


Figure 25: Final System CAD

5 Verification and Validation

Authors: Jordan Abell, Jett Moore, James Tiver, Nick Miller

5.1 Translator

5.1.1 Overview

The purpose of the translator tests was to ensure design requirements could be met before full system integration. Specifically the following design requirements were validated:

5.1.2 Accuracy Test

5.1.2.1 Purpose

To ensure design requirement could be met and to validate the uncertainty analysis performed in the fall semester. Based on this analysis the accuracy of the stepper motor/linear stage system is expected to be 1.13 mm based on the precision of each of these components. To validate this the translator offset was recorded at 5 locations with the encoder on and encoder off and compared to the expected location. This Test was run without the UR-10 connected to the base plate in order to match the assumptions made in the uncertainty analysis being validated. A summary of relevant design requirements is shown here:

DR	Source	Description
DR.7.2	FR.7	TThe translator shall be able to move to a prescribed location within a margin of 1 cm.
DR.7.1	FR.7	The translator shall be able to move to different positions along the track with a bag.

Table 3: Summary of Design Requirements Validated by the Translator Accuracy Test

5.1.2.2 Equipment

This test was conducted in the senior projects lab room using the stepper motor, linear stage, a laptop with an Ethernet port and a tape measure to record distances.

5.1.2.3 Procedure

1. Ensure no additional payload.
2. Open the SEM terminal in order to command the motor.
3. Zero Baseplate and record relative position.
4. Set maximum velocity to 358 RPM using the MDrive VM command.
5. Command motor to 5 locations ranging from 10cm-100cm.
6. At each location record the distance with the same reference as the origin using tape measure.
7. Use the EE=1 command to set the Encoder as the counter.
8. Repeat the previous steps with the encoder turned on.

5.1.2.4 Results

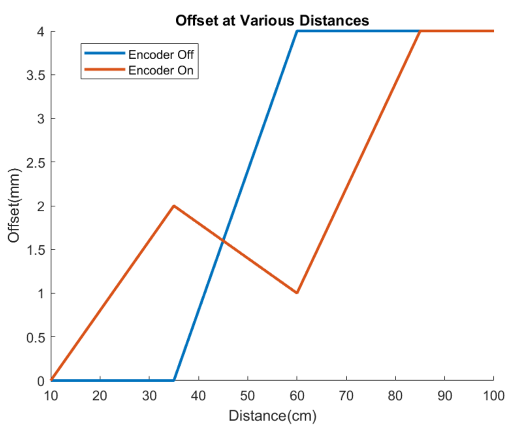


Figure 26: Translator offset at various distances with encoder positional counter enabled.

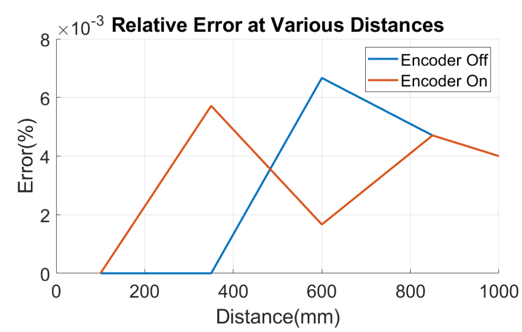


Figure 27: Relative error of translator positional offset.

The accuracy with the positional counter is similar to the encoder accuracy and both are accurate to within 4mm as seen on Figure 26. The design requirement is for 1 cm so these offset values are well within the designed range. At farther translation distances the offset slightly increases. This could be due to the counter having degrading accuracy, or human error of using the tape measure at longer distances. This human error could also contribute to the discrepancy between the uncertainty analysis and translator results. It can be seen that there isn't a clear trend in relative error on Figure 27 so the offset may just be proportional to distance. Although further testing could determine the cause of this error, it can be concluded from these initial measurements that the motor is easily accurate enough for the purpose of our system.

5.1.3 Mass Test

5.1.3.1 Purpose

To ensure that the system is able to validate design requirements concerned with transporting the mass of the system. This test was run prior to adding the UR-10 to ensure that no additional changes to the system would be needed before integration. This test was critical in ensuring the translator would be functional in the system. A slightly higher mass than the robotic arm was used for the final test in order to prevent unnecessary damage to the stepper motor and linear stage. The relevant design requirements can be seen here:

DR	Source	Description
DR.1.1	FR.1	The translation system shall be capable of translating the robotic arm, end-effector,and cargo bag's combined mass.
DR.7.2	FR.7	The translator shall be able to move to a prescribed location within a margin of 1 cm.

Table 4: Summary of Design Requirements Validated by the Translator Mass Test

5.1.3.2 Equipment

Linear stage, stepper motor, laptop, Arduino and a 26 kg and 60 kg mass.

5.1.3.3 Procedure

1. Zero the baseplate
2. Set maximum velocity to 358 RPM
3. Set motor to maximum torque mode
4. Set run current to 100%
5. Enable Encoder
6. Send command to translate to a relative position of 20 cm
7. Measure distance of travel and compare to expected distance
8. Increment weights on baseplate.

5.1.3.4 Results

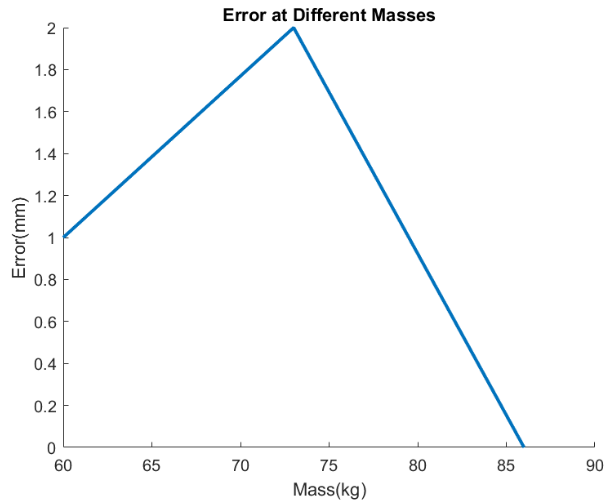


Figure 28: Offset for Mass Test run for 20 cm at 358 RPM.

At 20 cm the motor was accurately able to translate a mass that is 1.178 times the expected weight of the system. This is with zero errors and moderate speed. It can be seen in Figure 28 that there does not seem to be a correlation between mass and accuracy. Therefore, it can be concluded that the error was due to human error or inherent error in the system and that the translator is able to accurately move the UR-10.

5.1.4 Repeat-ability Test

5.1.4.1 Purpose

The purpose of the reputability test was to validate design requirements involving the repeated operation of the translator and time constraints. The relevant design requirements can be seen here:

DR	Source	Description
DR.2.1	FR.2	The system shall be capable of translating a cargo bag across the track in less than 10 minutes.
DR.2.3	FR.2	Translation across the track stall be repeatable within 1 cm of precision.

Table 5: Summary of Design Requirements Validated by the Translator Repeat-ability Test

5.1.4.2 Equipment

For measuring this test used a grid paper and stopwatch to validate the expected velocity commanded by the operator. An image of the grid paper can be seen on Figure 29This test was performed with the UR-10 attached so it also incorporated a 73 kg weight with the translator.

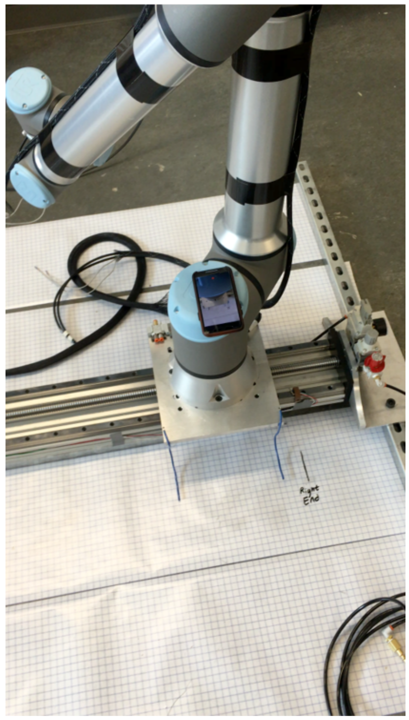


Figure 29: Testing Setup for Repeat-ability Tests. Grid paper used to measure offset.

5.1.4.3 Procedure

1. Zero the baseplate
2. Set maximum velocity to .0125 m/s for first trial
3. Set run current to 100%
4. Enable Encoder
5. Send command to translate to a relative position of 1 m
6. Measure distance of travel and compare to expected distance
7. Increment Velocity by .0125 m/s up to .05 m/s.

5.1.4.4 Results

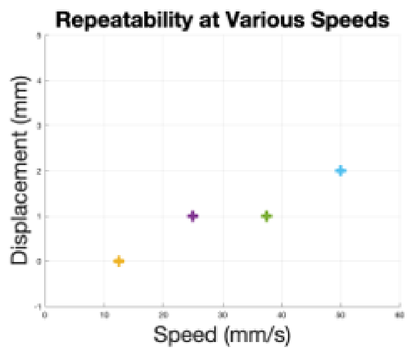


Figure 30: Results from repeat-ability tests at various speeds. Offset $<$ 2mm on all tests.

From this test it was found that the translation process was highly repeatable even at higher speeds. The precision from this test was even higher than the accuracy vs distance test. This most likely can be attributed to the use of grid paper instead of a tape measure providing more accurate measurements. By taking out the human error it was found that the translator is repeatable up to 2mm precision as seen on Figure 30 which is closer to the 1.13mm predicted by the uncertainty model discussed in the accuracy test section.

5.1.5 Holding Torque Model Validation

5.1.5.1 Purpose

To validate the holding torque calculated for CDR before procuring a stepper motor. In order to calculate the holding torque this equation was used to find the static force:

$$F_{static} = F_m + F_{f,static} = F_m + m * g * \mu_s = 729.7\ N \tag{4}$$

And this equation was used to convert the force to a torque:

$$T = \frac{F_a * l}{2\pi\eta} \tag{5}$$

From these calculations a holding torque of .6452 Nm was expected using an upper bound value for the coefficient of static friction.

5.1.5.2 Equipment

In order to validate this model the entire mass of the system was needed. Since the bag mass is negligible, the robotic arm with the end effector was translated using the linear stage and stepper motor using an Arduino connected to a laptop.

5.1.5.3 Procedure

1. Set run current to 100%

2. For initial test set Torque to 1%.
3. Enable Encoder
4. Set initial velocity to 0.
5. Increase torque by 1% until system begins to move.
6. Retest around torque value that allowed movement

5.1.5.4 Results

From these tests it was found that the stepper motor was able to move the system from standstill at about 22% of the total torque. Since the motor is rated to 1.71 Nm this means that the holding torque is found to be .3762 Nm. This is 58% of the predicted value of .6452, but this could be partially attributed to the value used for the coefficient of static friction. The value used was an upper bound of expected values. The only way to accurately obtain a coefficient of friction before integration, would be experimentation, but because our stepper motor had to be procured with our linear stage simultaneously these measurements were not able to be made. In order to develop a more accurate model for holding torque initial experiments on the linear stage friction could be used to obtain an accurate coefficient of static friction. In order to compensate for uncertainty in the model our team procured a motor with a much greater holding torque than our expected holding torque and used upper bound values when calculating holding torque.

5.2 End Effector

5.2.1 Overview

The purpose of the end effector tests was to characterize the ability of the end effector. This was done with tests that determined the maximum vertical and horizontal offsets in which the end effector would grab the bag. Another test was conducted to determine the maximum angle allowed at the maximum offset to grab the bag.

5.2.2 Margin Test

5.2.2.1 Purpose

The purpose of this test is to determine both the maximum vertical and horizontal offsets that the end effector is able to grab the magnetic plate on the bag. The tests were performed in order to determine the offset margin with an accuracy of 1mm.

5.2.2.2 Equipment

The only equipment necessary to conduct this test was the end effector, the robotic arm, the cargo bag, and a small ruler.

5.2.2.3 Procedure

1. Set the bag in a horizontal position with the magnetic plate facing up
2. Use the free-drive function on the UR-10e to place the end effector directly above the magnetic plate and center it.

3. Align the end effector with the Z-axis to ensure zero-angle attachment
4. Record coordinate position of the end effector
5. increase vertical offset by 1mm and actuate end effector
6. Reset bag position (if successful grab)
7. Repeat previous 2 steps until unsuccessful grab
8. Record maximum vertical offset
9. Set position of end effector to last successful grab position and move 1mm in the X-direction
10. actuate end effector, if unsuccessful grab move -1mm in Z-direction, actuate again
11. Repeat last step until successful grab, then reset bag for next trial
12. Repeat steps 9-11 until position of the end effector in the Z-direction is zero relative to the magnetic plate

5.2.2.4 Results

After recording the x-z locations of each maximum offset, we were able to create the following plot in Fig. 31. Figure 31 shows the end effector offset bounds, which helps inform the Software team of the required pointing accuracy of the end effector in order to successfully grab the bag.

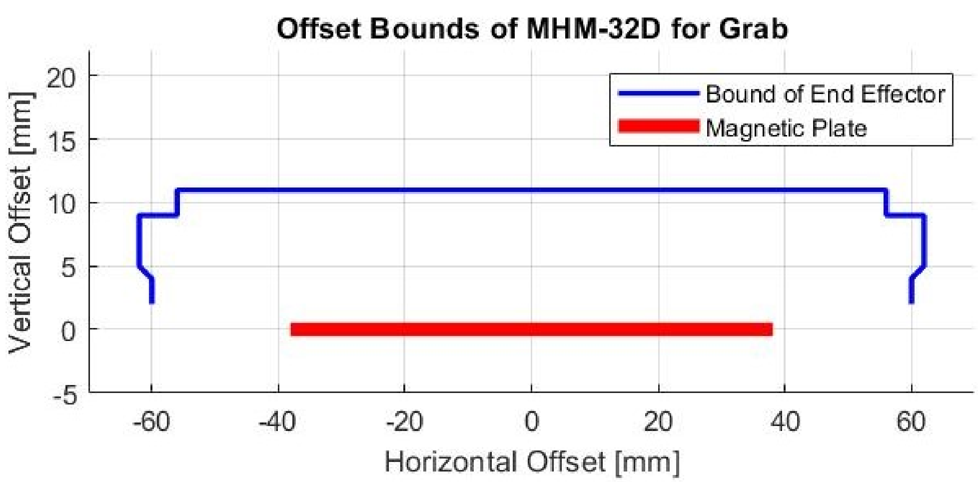


Figure 31

From Testing it was determined that the maximum vertical offset was 11m, while the maximum horizontal offset was 62mm. The maximum total offset from the center of the magnetic plate to the farthest point on the boundary was about 64mm. By tracing this plot within three dimensions it is clear that there is a region around the magnetic plate on the bag in which the end effector is effective in grabbing the bag for translation and rotation.

5.3 Software

5.3.1 Estimation Test

5.3.1.1 Purpose

The purpose of the Estimation Test is to verify that the end-to-end estimation process is capable of estimating the position and orientation of a cargo bag within the uncertainty bounds required by the end effector. The Estimation Test is made up of two subtests: the Camera Test and the Algorithm Test.

The Camera Test is used to validate the Monte Carlo Camera Visibility Model. The Visibility Model predicted that the sensor suite would be capable of obtaining necessary data for estimation in 89% of bag configurations.

The Algorithm Test is used to validate the Simulated Data Estimation Model. The Simulated Data Model predicted that the Estimation algorithms would be capable of estimating the positions of bag markers with an uncertainty (2σ bound) of 0.6 [mm].

5.3.1.2 Equipment

The following components of the overall system are necessary to conduct the Estimation Test: Computer, Pixy Sensors, End Effector, UR-10, Cargo Bag, tape measure.

5.3.1.3 Procedure

1. Set up the bag in a random position and orientation
2. Manually measure the center of the bag using a tape measure
3. Run the estimation filter using live sensor data
4. Record the estimate of the bag center
5. Plot the difference between the estimate and measured bag centers
6. Command the robotic arm to position the end effector at the magnetic plate
7. Engage the end effector, record if the bag was secured
8. Repeat for multiple trials and orientations

Four different bag orientations were tested for the Estimation Test.

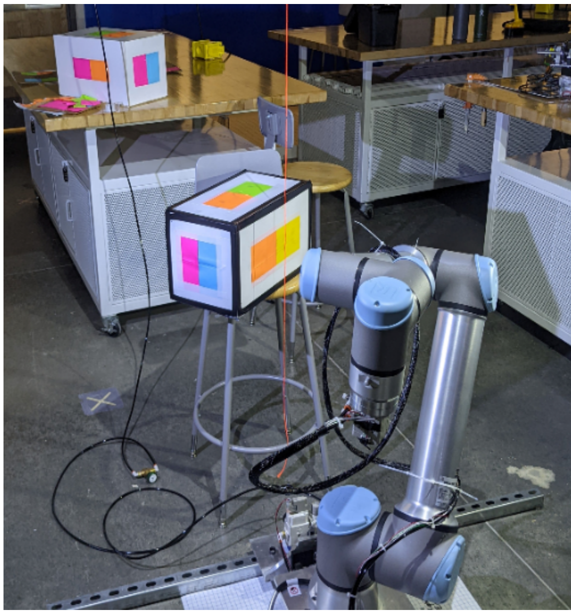


Figure 32: Estimation Test: Orientation 1

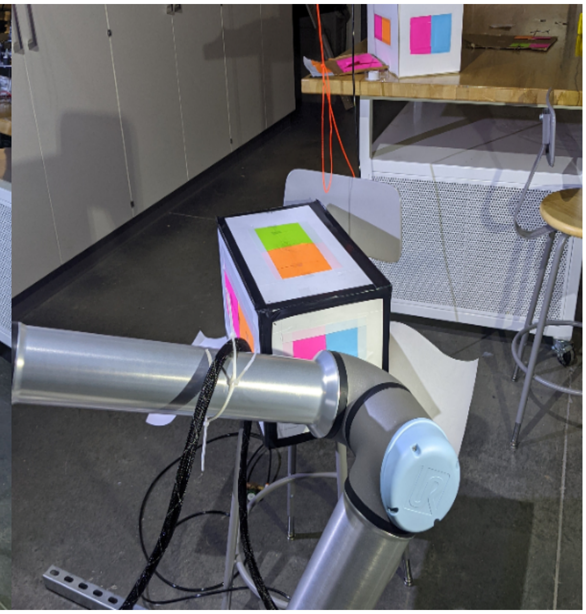


Figure 33: Estimtation Test: Orientation 2

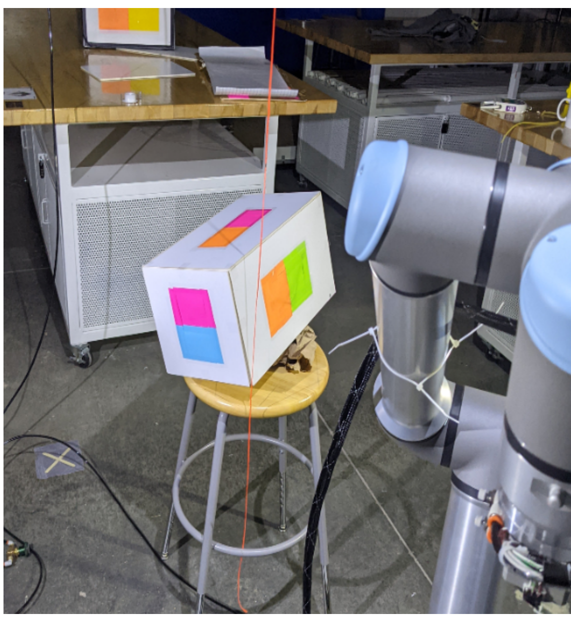


Figure 34: Estimation Test: Orientation 3

Figure 35: Estimtation Test: Orientation 4

Orientation 1, shown in Figure 32, was placed flat on the stool with a negative rotation about the $\hat{z}$ axis. The magnetic plate is under the marker on the top face.

Orientation 2, shown in Figure 33, was placed flat on the stool with a positive rotation about the $\hat{z}$ axis. The magnetic plate is under the marker on the top face.

Orientation 3, shown in Figure 34, was placed tilted on the stool with a complex rotation. The magnetic plate is under the marker on the front face.

Orientation 4, shown in Figure 35, was placed tilted on the stool along its long edge with a complex rotation. The magnetic plate is under the marker on the front face.

5.3.1.4 Results

5.3.1.4.1 Camera Test

Camera testing revealed that the physical Pixy2 sensors are not as robust in color and marker identification as previously thought. Through testing ranges of orientations to determine if the sensor suite was capable of collecting enough data for useful estimation, it was found that the acceptable range of rotation about the $\hat{Z}$ axis was approximately $\in \pm\{[20°, 70°], [110°, 160°]\}$. Similarly, the inclination (rotation about $\hat{Y}$ axis) must be in the approximate range $\in \{[-40°, 30°]\}$. Figure 36 defines the rotation definitions examined.

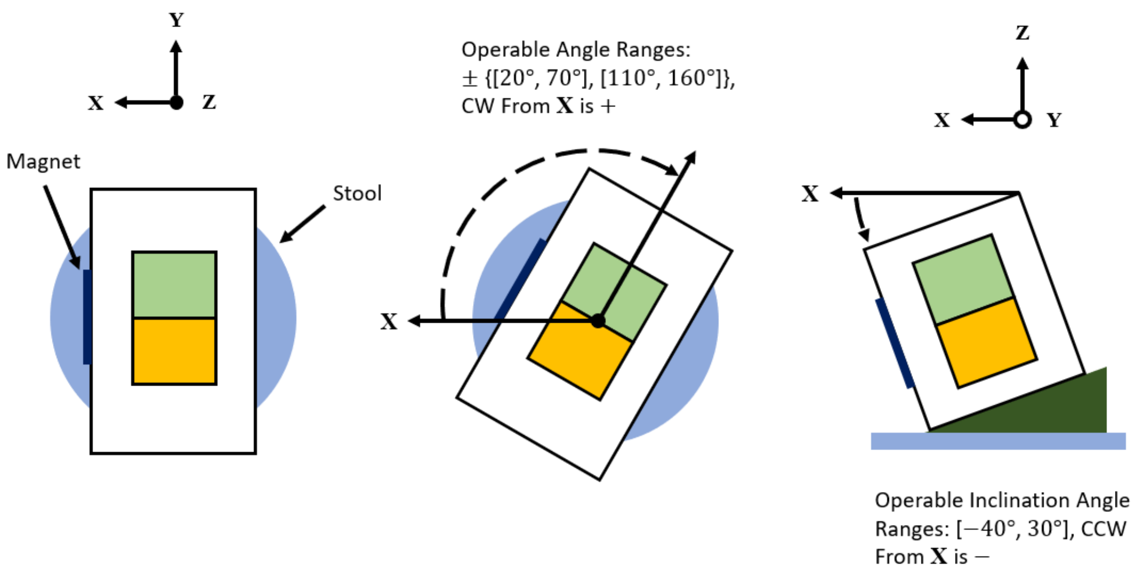


Figure 36: Rotation Definitions

These results are difficult to quantify as a percentage of orientations, but it is safe to say that it does not equate to 90%.

While this does not match our previous model, the orientations that are favorable to estimation are purely a function of the arrangement of the sensors. Changing the positioning of the sensors and adding sensors to a designated area will both drastically change the orientations that can be estimated. This is a portion of the project that could use significantly more analysis to determine the optimal arrangement of sensors. One possible analysis method to examine is to use the Fisher Information Matrix to determine arrangements that provide the most information for given orientations.

5.3.1.4.2 Algorithm Test

Figure 37 shows the results from the Algorithm Testing of the error between the estimated and the measured bag center.

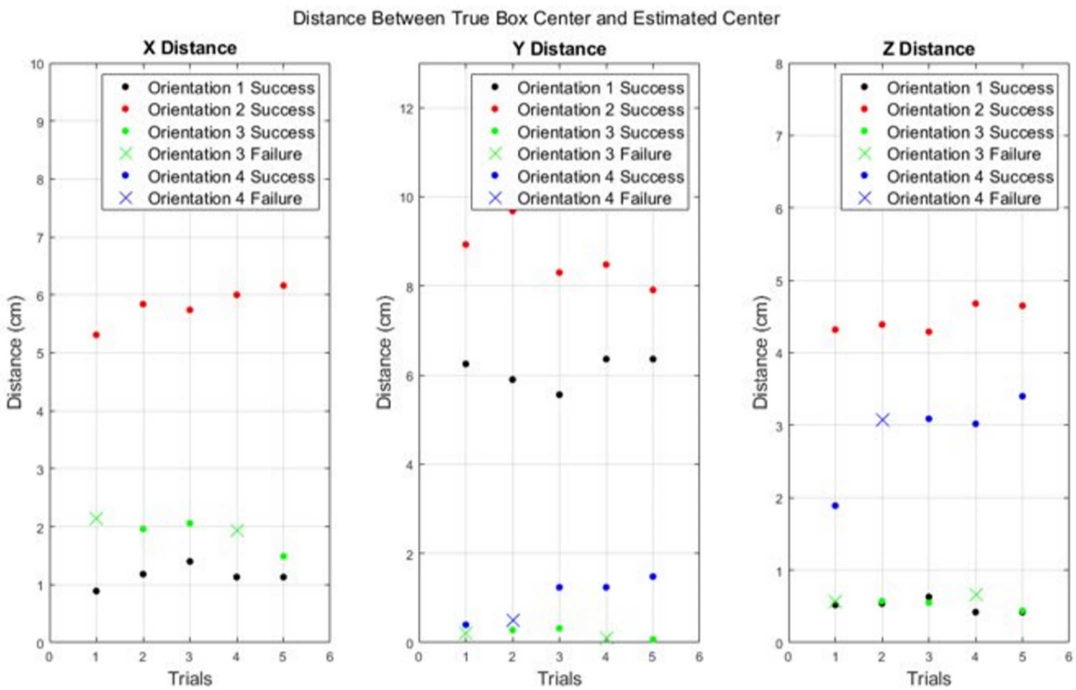


Figure 37: Results from Algorithm Testing

There are many important takeaways from these results.

First, for each orientation tested, the errors are relatively consistent across each trial.

Next, while errors for some orientations are on the order of 10 [cm], the system was still successful in securing the bag. This indicates that the system and the estimation software are consistent. Further, this indicates that the cause of the large error must be external to the system, so it must be due to the measured box center. One possible source of this error is human error in the measurement. This is likely responsible for some of the error due to the fact that manually measuring the bag's center is difficult when the bag is in complex orientations. However, it is unlikely that human error is responsible for errors of this magnitude. Another likely error source is that the measurements were taken from an inconsistent reference point. This means that the origin that is physically defined in the test environment is not the same origin that is computed in the estimation software. A good way to diagnose this error would be to place a marker at the physically defined origin and examine whether the estimation software also thinks that the marker is at the origin.

Another significant takeaway is that there is no correlation between bag center estimation error and system failure to secure the bag. This indicates that estimation error that is present in the system does not impact the system performance enough to inhibit the system.

5.3.2 ROS Environment Test

5.3.2.1 Overview/Purpose

The purpose of the ROS Environment test is to verify that each individual ROS Node sends and receives the correct data at the correct time. Since the ROS Environment consists entirely of software to software interfaces (with the exclusion of ROS Serial and UR10e driver packages), the tests were evaluated on a pass/fail basis.

5.3.2.2 Equipment

In order to conduct the entire ROS Environment Test, all components are needed. Specifically the equipment needed used was, the Linux Laptop. the camera Arduino, the motor and sensory Arduino, the suite of Pixy 2 cameras, the motor, the robotic arm, all end effector components, and various sensors.

5.3.2.3 Procedure

1. Start *roscore*.
2. Run various publishing nodes.
3. Run subscribing test nodes.
4. Use *rostopic list* to verify that the ROS topic exists.
5. Verify the data is received by the subscribing nodes using terminal output statements.
6. Compare received data to the data being published.
7. If the communication uses a ROS service, verify that the service is being provided using *rosservice list*, then verify that the service receives the correct data, and sends the correct response.

5.3.2.4 Results

Each individual pair of communicating nodes was tested during development. Once all of the necessary combinations were verified using terminal print statements, node integration tests were performed. This involved testing more than two communicating nodes at once to verify that there were no call back function conflicts, or service conflicts. These tests were also performed during development. The entire ROS environment test was performed during the Full Systems test. During operation, the task planner execution output was monitored by the user, who could verify that each hardware interface command was properly executed. During all full system test trials, it was found that each software to software ROS communication interface was working correctly by verifying the terminal output from each individual node after being commanded by either the Task Planner Node or the Status Node. It was also found that each software to hardware interface was working correctly by visually verifying the hardware command, or sensory input.

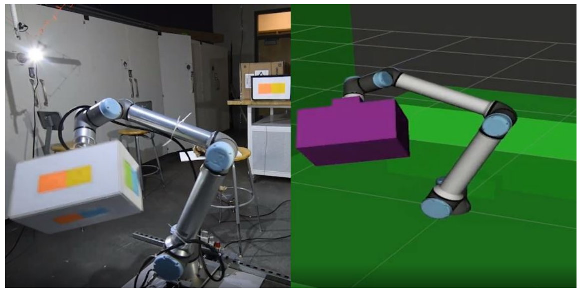


Figure 38: Real vs Simulated Motion Plan Validation

Figure 38 shows a side by side comparison taken at the same instant in time of the real motion plan execution versus the simulated motion plan execution (in RViz). The user can easily verify that the motion being planned for in simulation is the same motion being executed by the controllers on the robotic arm. All hardware components were successfully validated by the user during the Full Systems Test.

5.4 Full System

5.4.1 Overview

The purpose of the full systems testing is to quantify results regarding the entire integrated system. The testing will evaluate how repeatable and reliable the system is.

5.4.2 Repeatability Testing

5.4.2.1 Purpose

Repeatability testing is required to ensure the various components working together can successfully accomplish the mission of transferring cargo. The testing will give a success percentage rating of the design, but observations will also be made to detail which subsystem or component may be bottle necking the performance of the system. This test also acts as a second confirmation of every design requirement the subsystems tested, but again with the rest of the subcomponents.

5.4.2.2 Equipment

The test was completed in the projects lab room using all components of the project. A summary of the major components include, the robotic arm, the Pixy 2 cameras, the limit switches, the stepper motor, the linear stage, the end effector, the estimation and communication software and the laptop running the system. Since the end effector was used, the pressurized air from the senior projects room was also utilized.

5.4.2.3 Procedure

1. Ensure all components are powered on and wired to the Arduinos and the Laptop.

2. Initiate the system in ROS.

3. Check position of cameras and calibrate their positions.

4. Pixy 2 cameras scan the environment for the cargo bag
5. Estimation algorithm triangulates cargo from camera's pixel data
6. Motion planner queues trajectory for the robotic arm to capture the cargo
7. The robotic arm captures the cargo then moves to a safe pose, and then the translator moves to the drop off zone.
8. The robotic arm then drops off the cargo and moves back into a safe pose to move back to the pick up zone to restart the operation.

5.4.2.4 Results

Due to limited time at the end of semester, only 22 trials were tested for the repeatability test. At the end of testing 17 trials were successful from start to end giving the system a 77.27%. Four different orientations were tested starting with simple flat orientations and escalating to non simple slanted rotations. Each orientation was tested five times. Then a fifth orientation was tested twice but failed both times. The system failed to plan a trajectory for the arm to the bag due to an error with the Pixy 2 cameras. It was believed that one of the cameras may have been mismatching colors which caused the estimation algorithm to misplace a marker. This caused the orientation of the box to be floating off of the surface and orientated in a position where the arm thought it wouldn't be capable of capturing the bag.

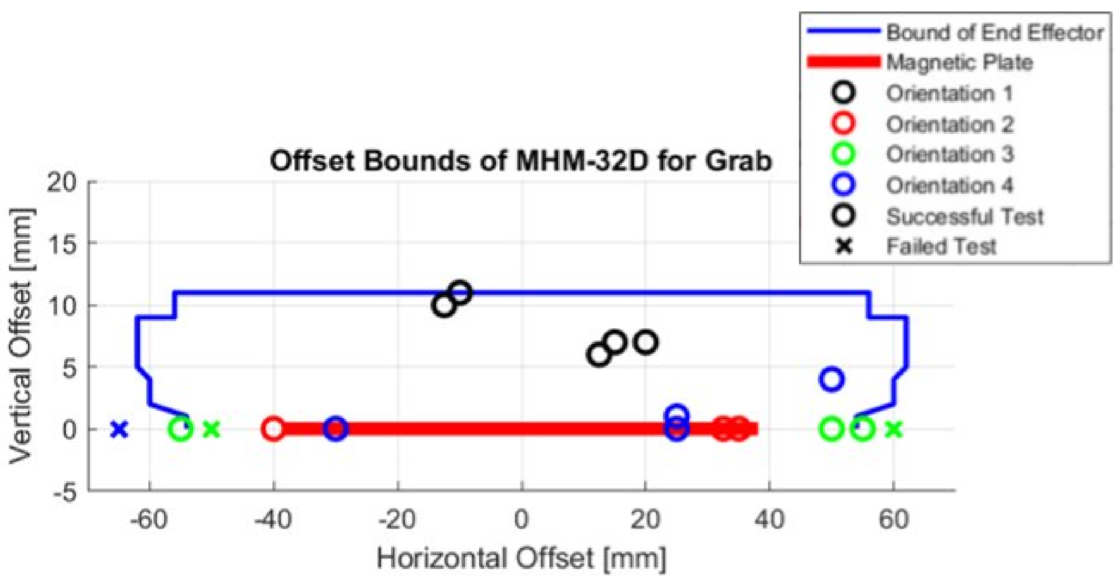


Figure 39: System Test Results with End Effector Bounds

Figure 39 shows the results of each orientation trial overlaid on the End Effector bounds necessary to secure the bag. The offsets are measured from the center of the End Effector to the center of the magnetic plate. In all but four of the trials the End Effector was placed within the required bounds to secure the cargo bag, which indicates that the overall system is largely working as designed. Two of the four trials that were outside the bounds were unsuccessful, as would be expected. There are a few possible reasons that the other two trials were still successful. First, the offset measurements could have been slightly mis-measured, rounding them outside the End Effector bounds. Additionally, the End Effector bounds could be slightly conservative, and successful capture is possible just outside of the bounds.

One trial that was placed inside the bounds was unsuccessful. This trial was noted to have an unusually large angular displacement between the End Effector and the magnetic plate. This displacement is difficult to measure accurately, but we can attribute this as the cause of the failed trial.

Possible sources of error in these tests include the camera system losing their calibration position. When the sensor cage is unintentionally bumped, the cameras slightly change their pointing angle and can cause discrepancies in the estimation. It's believed that if a more stable mounting solution is used then the offset errors can be reduced.

From the results of the full system test and the additional validation test there are a handful of improvements that we would recommend to improve overall system performance. First, developing a custom color/marker recognition algorithm would allow for greater flexibility in sensing. This would allow the sensors to more reliably identify cargo bags for capture. Next, improving the quality of the sensor mounting system would drastically improve the estimation performance. The position estimation is particularly sensitive to the pointing of each sensor, and a higher-quality mounting solution would help decrease uncertainty in the sensor pointing. One possible solution may be ball and socket joints that allow for more precise adjustments. Finally, a higher fidelity test environment would allow the overall system to better reflect the intended conditions for operation. This would significantly improve our ability to draw conclusions that accurately reflect true operating performance. These changes would all make the system more repeatable and robust, bringing the design closer to being capable of transitioning to a true flight system.

6 Risk Assessment and Mitigation

Authors: Lindsay Cobb

6.1 Risk Identification and Tracking

A potential risk to the project was identified as any series of events that could seriously endanger the success of the project or prevent the satisfaction of requirements. Risks were measured by the likelihood of their occurrence the severity of the consequences. This applies to hardware, software, schedule, and cost of the project. Risks were identified during the design phase as anticipated events that may occur unless properly mitigated. During the manufacturing and testing phases, newly identified risks were presented weekly by individual sub-team. The risks that were most critical to the overall success of the project were added to the tracking list and efforts were made to find a mitigation strategy. Figure 40 shows a risk matrix with the 7 most critical risks of the project. The full list of project risks is included in Table 6

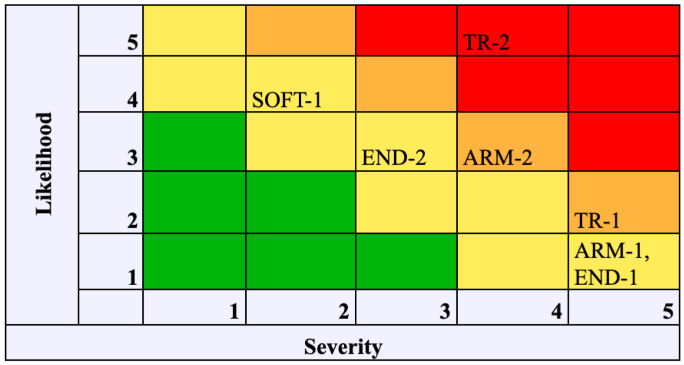


Figure 40: Top Priority Risk Matrix

ID	Risk	Severity	Likelihood
ARM-1	Arm is commanded to a position that damages it	5	1
ARM-2	Arm cannot make contact with bag	4	3
TR-1	Translator threads are stripped	5	2
TR-2	Linear stage tipping over	4	5
SOFT-1	Cargo bag cannot be identified	2	4
SOFT-2	Camera system accuracy cannot locate bag within required margin for the end effector	4	3
SOFT-3	Drop off and pick up location cameras cannot work in tandem	2	5
END-1	Pressure actuator failure	5	1
END-2	Improper alignment/bag dropping	3	3
MGT-1	Project schedules delays full system testing	4	3
MGT-2	Project cost is over budget	5	4

Table 6: Complete List of Risks

6.2 Risk Mitigation

ARM-1 The UR10e Robotic Arm has a built in safety feature that stops the arm in place if it collides with an object. This will prevent the arm from being damaged if it runs into an obstacle in the environment or itself.

ARM-2 If the arm cannot make contact with the bag, an operator can manually control the position until it is close enough to capture the bag.

TR-1 A limit switch was implemented on either side of the translator track so when the base of the arm actuates the switch, the translator motor shuts off. This will prevent unwanted torque or jamming of the drive screw that could strip it's threads.

TR-2 Beams were added on either end of the translator that lie perpendicular to the track. They prevent the stage from tipping over if the arm were to reach to far to either side. These beams are in place of physically securing the translator to the floor which was not an option for testing.

SOFT-1 If the cargo bag cannot be identified with the camera sensors, an operator can manually control the arm to the desired position to capture a bag.

SOFT-2 If the camera system is not accurate enough to capture the bag, the cameras can be brought closer to the staging area, more lighting can be added, and the size of the markers can be increased.

SOFT-3 If the 5 drop off and 5 pick up cameras cannot work in sync, then only 5 at a time will be used. First the pick up cameras will identify a bag and then the drop off cameras will verify the bag has been placed correctly.

END-1 In the event of a pressure actuator failure, a new part will be swapped out.

END-2 If the alignment of the bag and end effector is incorrect and leads to dropping the bag, the size of the steel plate on the bag can be increased.

MGT-1 There is no mitigation plan if the schedule delays exceed the deadline.

MGT-2 If the project cost exceeds the budget, components will be returned and the project will be descoped to stay under budget.

6.3 Risk Impact

Risks SOFT-1, SOFT-2, SOFT-3, and MGT-1 were the risks that were realized during the course of the project. All other risks were sufficiently mitigated. When the camera systems could not identify the bag, the position of the bag had to be manually input so the arm could properly capture it. This problem was solved by improving the lighting so the cameras could consistently identify the ID tags on the bag. The orientation had a significant impact on the cameras' ability to identify the bag with 4/5 orientations being successful. The camera system accuracy was typically a systematic bias that could be accounted for by manually adding or taking away margin from the positioning of the robotic arm. The drop off location cameras did not work in tandem with the pick up cameras which was not resolved during testing. This was due to schedule delays that did not leave time to test and debug the drop off cameras. While a full systems test was complete, schedule back up prevented thorough and repeated testing from being run. The impact of these problems primarily affected the schedule and delayed further testing, however the project still reached level 4 success for all categories.

7 Project Planning

Authors: Kyle Li, Alex Ferguson

The potential for delays in the spring semester was a major risk to the potential completion of the project's objectives. These delays include shipping issues, fabrication errors, and debugging difficulties. To further ensure that delays did not occur project management was approached with three major ideas: transparency, communication, and coordination.

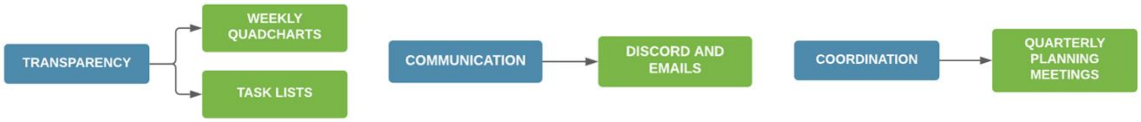


Figure 41: Project Management Approach and Solutions

From the figure above it can be seen that transparency was ensured for the group as a whole by incorporating weekly quadcharts where each subteam, including the administrative team, presented on that week's tasks and status updates. In addition, tasks lists were utilized where all members of the team were assigned to perform specific tasks by a determined due date to advance the progress on deliverables, tests, and internal deadlines. All communication within the team was done within Discord with administrative emails forwarded to all subteam leaders. Discord also served as the central location for messaging, low-memory file sharing, and link sharing. This helped ensure that all team members will not miss important announcements or information. Finally, coordination among the various subteams was improved by quarterly planning meetings where the leaders of the various subteams meet to plan out expectations, resource allocations, and discuss major risks associated with the project at the time. With all these steps in place, the project sought to minimize lull and crunch periods while preventing the occurrence of issues.

The team is divided between three major teams: structures, electronics, and software/sensing. Each member of the team is responsible for an important component of the project that ranges from testing lead to electrical lead. There will be members of the team that interact with more than one of the critical project elements over the course of the project and their focus is reflected on the chart below:

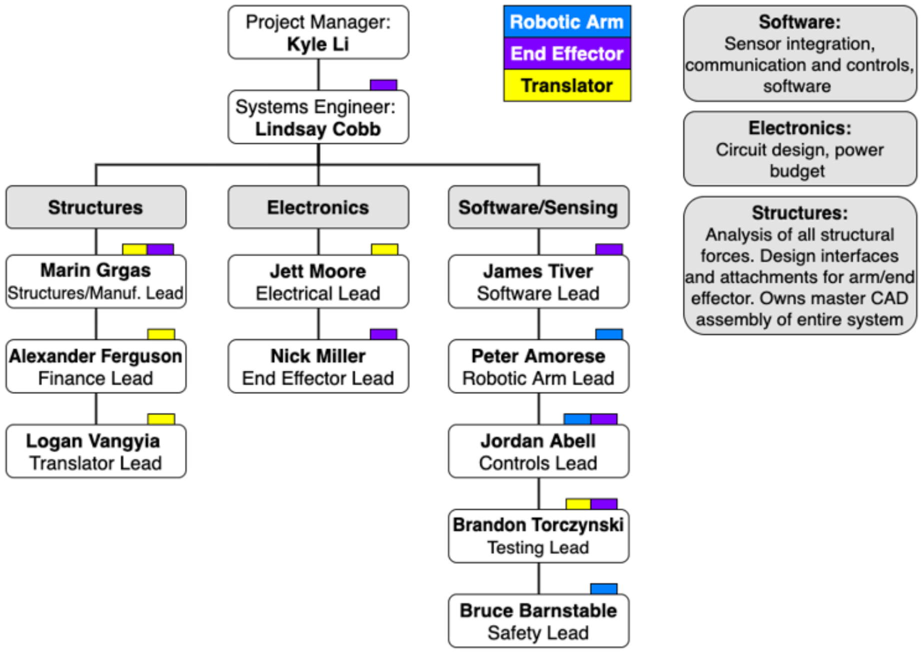


Figure 42: Organizational Chart

7.1 Work Breakdown Structure

The work breakdown structure featured below shows the planned major objectives that have been completed by the end of this semester. These major future objectives are focused on in a detailed work plan shown later in this section. The component with the most tasks is the translator mostly in part due to the complexities of the system and its need to integrate with the rest of the system. However, the project's most time intensive component was actually the robot arm and integration with ROS.

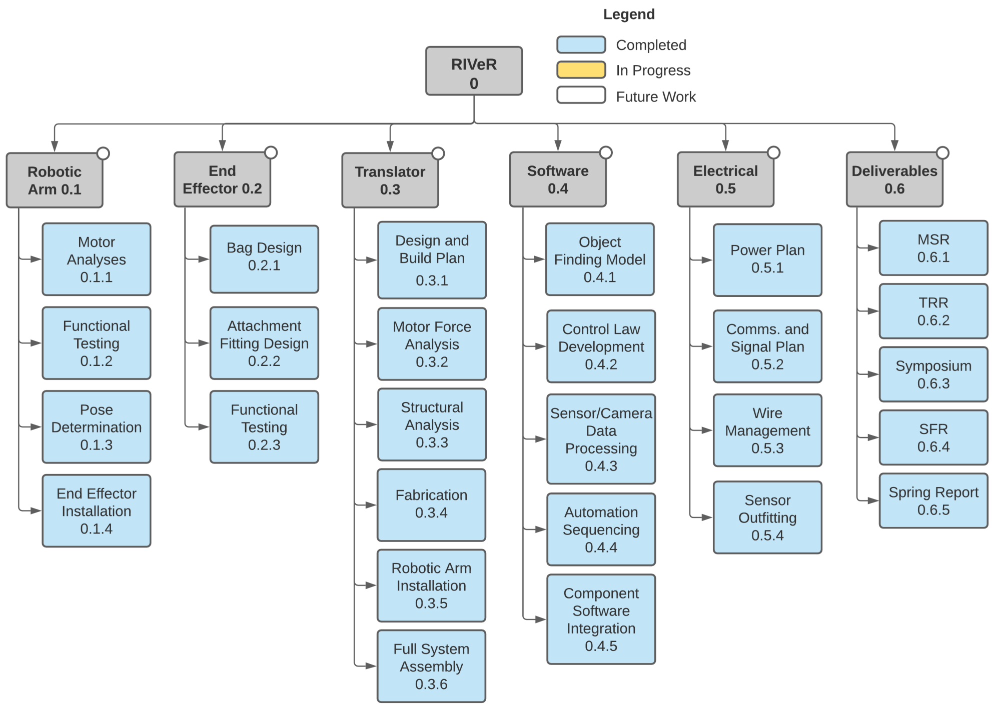


Figure 43: Work Breakdown Structure

7.2 Work Plan

The work plan shown below is a summarized variant of the actual work plan. The actual work plan can be found in the appendix with all the tasks associated for each major category. The project is divided into three major phases. Phase I is 5 weeks long and Phase II and III are 4 weeks each. Each stage has an integration phase in addition to fabrication, software, and testing phases that encapsulate all major objectives that need to be done in that category. Each phase has a major milestone which builds up on all the fabrication, software, and testing work done earlier in that phase. Phase I has the stationary test which requires the robotic arm to reach the bag at a specified location, activate the end-effector, and finally, translate the bag from the starting area to an area behind the arm. Once this milestone is complete the fabrication of the translator can take place in Phase II. This fabrication process will involve the integrating the stepper motor, the base plate, the rail system, and the controls of the translator prior to final integration with the arm. Once this has taken place the dynamics test will focus on integrating commands for the translator and the robotic arm from the PC to pick and place the bag from one end of the translator to the other. In Phase III, the Pixy camera integration will take place. The bag's location will be incorporated to triangulate the bag's position and adapt the robotic arm's pick up operation. The full systems test will focus on the arm's ability to adapt to different bag placement locations and perform one cycle of the translation operations. Throughout all three phases ROS functionality will be progressively improved and incorporated over time to build upon the arm's functionality and capabilities.

Over the course of the project in the spring there was a major delay associated with the stationary and dynamics test, where both tests were delayed by a week. Furthermore, during the full systems test, functionality was only established the day before and the day of the Symposium on April 15-16th. Fortunately, all major levels of success were accomplished within the project's lifespan.

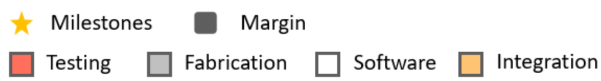


Figure 44: Gantt chart key

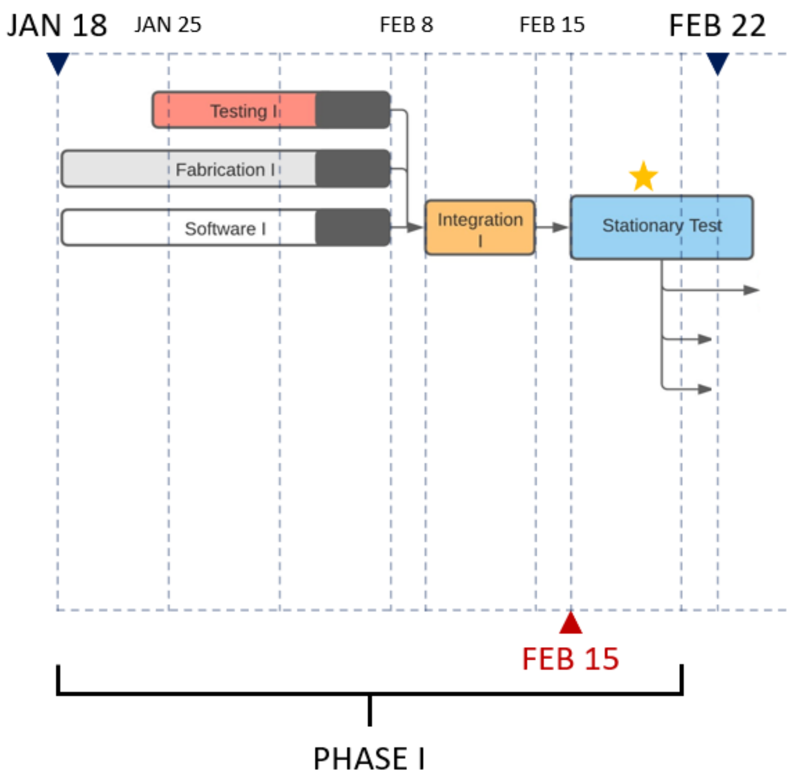


Figure 45: Phase I of work plan

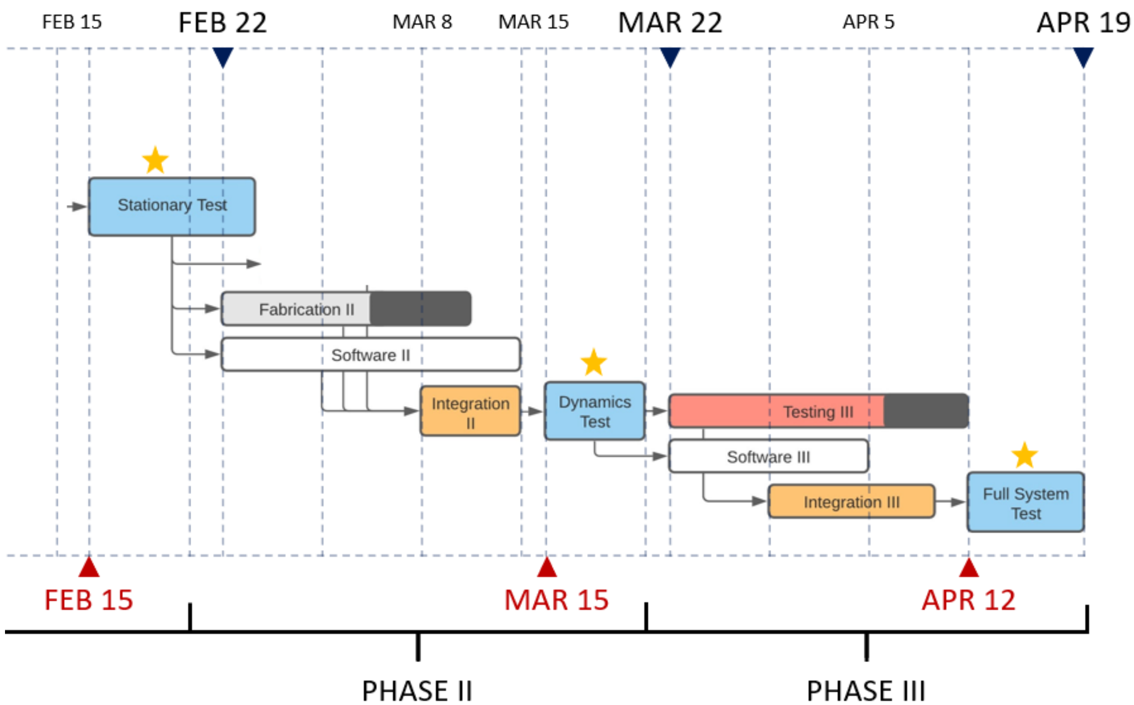


Figure 46: Phases II and III of work plan

7.3 Cost Plan

The cost plan shows that the project ended up being $182.13 under-budget. The primary expenses for the project come from the translator, followed by the end effector, and lastly the sensor suite. The cost of the project increased by $1,114.67, a significant increase, over the course of the spring semester as seen in the cost plan table (7). It is important to note that the robot arm, the UR10e, was given to the team for free on a loan for this project.

Subsystem	CDR Cost	Finalized Cost	Overage	Allocated Amount	Budget Margin
Translator	$1,785.80	$2,168.56	$382.76	$2,200	$31.44
End Effector	$998.17	$1,354.32	$366.88	$1,600	$245.68
Sensor	$728.11	$1,094.99	$365.03	$1,000	– $94.99
RIVeR Total	$3,512.08	**$4,626.75**	**$1,105.59**	**$4,800**	**$173.25**

Table 7: Cost Plan

The figure below outlines the biggest expenses for the project (47). The largest expenses for the translator subsystem were the linear stage and NEMA 23 motor. These two items were responsible for over 60% of the subsystem's costs. The most expensive items for the end effector subsystem include the magnetic gripper, dump valves, and the UR10 base-plate. These three items made up over 60% of the subsystem budget. Lastly, the big ticket item for the sensor subsystem was the 10

Pixy2 cameras, which accounted for over half of the subsystem costs.

The additional purchases post-CDR were due to a myriad of reasons. The figure below (48) shows an itemized list of the purchases made over the spring semester. The translator subsystem incurred new costs. Highlights include a coupler to connect the motor shaft to the linear stage, bracket mounts for the motor, and a cable carrier. Additionally, a temporary mounting solution was requested by the Sierra Nevada Corporation as a precaution for the robotic arm. This led to us needing to purchase struts and hex screws to harness the arm (48).

The end effector subsystem also required additional purchases during the spring semester. Big ticket items include a base-plate to connect the robotic arm to the linear stage, actuators for the end effector, and a driver for the system's motor. (48)

Lastly, the sensor subsystem incurred additional costs over the semester. This included creating camera mounts with PVC pipes, wood planks, and corner braces. Furthermore, LED lights were needed to illuminate the test area so the cameras could better recognize the color scheme on the cargo bags. There were also a plethora of cables and wires needed to integrate the sensor suite with the entire system (48).

Translator	Total Subsystem Budget	Linear Stage	Motor	Al 6061 1/2"	Cable Carrier	Struts (x4)	Total Big Ticket
	$2,168.56	$749.99	$660.45	$226.18	108.84	160.93	$1,906.39
Budget Percentage		34.58%	30.46%	10.43%	5.02%	7.42%	
End Effector	Total Subsystem Budget	Magnetic Gripper	Dump Valves	Solid State Sensors	Actuation System	UR10 Baseplate	Total Big Ticket
	$1,363.20	$491.45	$182.50	$89.24	$94.46	$177.15	$1,034.80
Budget Percentage		36.05%	13.39%	6.55%	6.93%	13.00%	
Sensors	Total Subsystem Budget	Pixy Cameras (x10)	Camera Mounts	Arduino Uno (x2)	Camera Test Stands		Total Big Ticket
	$1,094.99	$550.00	$48.85	$49.99	$94.09		$742.93
Budget Percentage		50.23%	4.46%	4.57%	8.59%		

Figure 47: Big Ticket Items

Translator Items	Cost	End Effector Items	Cost	Sensor Items	Cost
Shaft Coupling	$26.62	Stepper Driver	$22.94	High-Pressure Pipe	$13.41
Cable Carrier	$108.84	Manual Dump Valve Coupler	$5.49	PVC Pipe	$37.98
Cable Mounting Brackets	$8.44	UR10 Baseplate	$177.15	PVC Pipe Fitting	$14.85
Strut Channel - Zinc Plate Steel	$160.93	Micro Switch for Bag Detection	$17.63	RAM Swivel Socket	$74.91
Hex Head Screw (x5)	$10.20	Micro Switch Connection Wires	$4.95	1-1/4" Phillip Screw Pack	$7.86
Hex Nut (x5)	$4.73	Zinc Head Screw- 10 mm (x50)	$15.53	Arduino Motor Shield	$13.49
Grid Paper Roll	$16.99	Steel Head Screw- 14mm (x25)	$5.73	PLTC Cable	$14.64
Steel Head Hex - 1.75" (x5)	$33.02	Brass Industrial Quick Disconnect Hose Coupling for Air	$3.24	Belkin Ethernet Cable	$7.42
NEMA 23 Mount Bracket	$12.99	M8 3-Pin Connector	$9.99	2x4 x 8ft	$18.45
		Telemecanique M8 Sensors	$25.74	2x4 x 10ft	$9.62
		Limit Switch	$6.12	Corner Braces	$7.72
		Solid Inline PNP Direct	$61.64	Nylon Mason Line	$5.47
				15 ft Micro USB Cable	$45.12
				LED 100W Chip (x4)	$39.86
				Protoboard	$9.94
				Voltage Booster (x2)	$8.99
				White Foam Board (x10)	$31.98
				Matte Spray Black	$4.97
$382.76		**$356.15**		**$366.68**	
		$1,105.59			

Figure 48: Post-CDR Purchases

In summary, the project will feature three major phases of development. These phases will generally include a testing, software, and fabrication component. A major milestone is present for each phase that must be completed prior to the advancement to a future phase. This planned schedule is outlined in the work plan. All of the team's members have a duty directly related to testing, software, fabrication, and/or integration for the project outlined in the organizational chart. There are specific tasks/objectives that must be completed for each testing, software, fabrication, and integration step outlined in the work breakdown structure. The cost plan has outlined the team's overall financial stability and shown that this project still has a large margin of wiggle room to adjust plans and account for unexpected expenses as necessary in the Spring semester.

7.4 Test Plan

The test plan table is displayed below:

Testing Plan I

Test Name	Definition/ Purpose	Driving Requirements	Completion Date	Location and Equipment
Robotic Arm: Functionality Test	Test UR10e Motion Planning and Pose Determination.	FR.1 FR.2 FR.4	February 5th	Senior Projects Room Wall Power Supply
End-Effector: Static Load Test	Test Capture Mechanism and End-Effector Controls with Bags.	FR.1 FR.2 FR.5 FR.6	February 5th	Senior Projects Room Air Supply Wall Power Supply
Translator: Calibration and Arduino Test	Test Motor Accuracy, Sensor Functionality, and Translator Controls.	FR.1 FR.7 FR.8	February 19th	Senior Projects Room Wall Power Supply
Pixy Camera: Bag Identification Test	Test Camera Triangulation Method and Accuracy for Bag Retrieval.	FR.1 FR.3 FR.4	March 12th	Senior Projects Room Wall Power Supply
Robotic Arm: ROS Test	Utilize ROS to Integrate the Translator and Arm for Dynamics Test and Failure Detection.	FR.3	March 19th	Senior Projects Room Wall Power Supply

Figure 49: Test Plan Part I

Testing Plan II

Test Name	Definition/ Purpose	Driving Requirements	Completion Date	Location and Equipment
Cargo Bag: Varied Bag Fidelity for Capture	Test Capture/Identification of Cargo Bags of Varied Fidelity	FR.1 FR.2 FR.3	April 9th	Senior Projects Room Air Supply Wall Power Supply
Full System Software Test	Test Control Software of the UR10, Translator, and Camera System with Single PC Control.	FR.1 FR.2 FR.3 FR.4 FR.5 FR.6 FR.7 FR.8	April 9th	Senior Projects Room Air Supply Wall Power Supply
Full System Functionality Test	Test UR10, End-Effector, Translator, and Camera System for a Full Cycle of Operations.	FR.1 FR.2 FR.3 FR.4 FR.5 FR.6 FR.7 FR.8	April 9th	Senior Project Room Air Supply Wall Power Supply
Full Systems Test	Test Software and Hardware of the Assembled System for a Full Cycle of Operations.	FR.1 FR.2 FR.3 FR.4 FR.5 FR.6 FR.7 FR.8	April 16th	Senior Projects Room Air Supply Wall Power Supply

Figure 50: Test Plan Part II

All major equipment that was needed in the test was acquired independently by RIVeR from a vendor, obtained via permission from the machine shop or electronics shop, or was donated by a team member for this project's use. For our tests in Testing Plan II, a DC power supply was needed to operate the lights which was borrowed from Trudy Schwartz with her permission.

8 Lessons Learned

Authors: Jett Moore, James Tiver, Logan Vangyia, Kyle Li

8.1 Translator

The major takeaways from the translator subsystem were related to designing parts to fit together. When designing parts during first semester, measurements were based off of CAD files and drawings found on the parts we were purchasing, but some of them were not correct so designs needed to be adjusted at the beginning of the second semester once parts had shipped to the senior projects room. The lesson learned from this mistake is to request more elaborate design drawings. The other lesson learned from this sub-team was how hard tolerances are to regulate when making screw holes line up. When making a moving platform, friction needs to be minimized which any amount of misalignment can ruin. The solution to this issue was turning some of the holes into slots that can be easily readjusted to fit the system in the event of slight misalignment or warping of materials.

8.2 Electronics

There were several takeaways concerning electronics regarding interfaces and ordering. Firstly, the wiring for the Pixy2 camera's was by far the section of electronics that had the most issues. The T2 connectors initially used for the I2C lines were difficult to use and often times failed a preventing the wires from shorting. Therefore, for future electronics projects it is recommended to use the standard twist on wire connectors, or some other solution. Twist on wire connectors worked without any issues for the team and should be used for wire connections in the future. Additionally, the wire used to rout the I2C camera's is solid rather than signal wires. Although there have been no issues so far, if the wire is subjected to bending in the future there could eventually be failure. Therefore, for future long distance wiring it is recommended to use signal and power cables that involve wire bundles, rather than a solid wire, to increase durability. Additionally, even though wire lengths were measured and excessive wire was ordered, the team still needed to procure additional wire from the electronics shop. Considering the price of wire, it is recommended to procure bulk wire in initial purchasing. Finally, it was found that thermal considerations need to be made for high power electronics components. The solenoid driver overheated and the LED's needed thermal relief before they could be mounted to the sensor cage. A thermal analysis in the fall semester would have better prepared the team for mounting the electronics in the spring. Another lesson learned is that most of the Arduino shield pins cannot be used. Some of them will only partially impair the functionality of the system which led to a lot of debugging. The digital ports that can be used on the Arduino are 5,6,7,8,9,16 and 17. For further reference look at the Arduino Ethernet V2 schematic shown in Figure 51. Finally, there was an issue with the first USB hub. The team prioritized cost due to budget constraints and ended up with a power supply that had too much noise for the I2C lines to function. From this the team learned how important ordering parts with a good data sheet and low noise if they are supplying a regulated power.

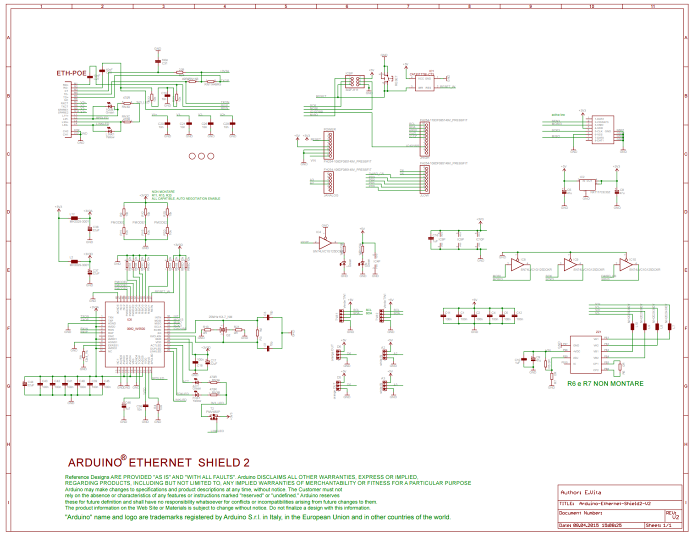


Figure 51: Schematic of The Ethernet Shield V2 for Arduino. Use when attempting to plug additional I/O pins to avoid interfering with the Ethernet SPI conversion.

8.3 End Effector

The process of integrating and testing the end effector was relatively seamless. Our testing showed that the end effector has a high offset margin between the center of the magnet and the center of the steel plate. This made it easier to orient the arm in way where it could engage and disengage with the box than initially expected. This in conjunction with a larger plate diameter, simplified the software team's motion planning algorithm. There were, however, a few difficulties. The first problem we encountered was not having a predetermined wire management scheme. This meant having to solder wires together to increase the length as well as having a messy electronic work space that made adjustments tedious and frustrating. In hindsight, we should have determined where would place our components and then figured out an optimal wiring scheme. However, we made due with zip-ties and cable track, minimizing some of the frustrations. The next obstacle was the solenoid driver burning out. This could have been prevented fairly easily by having the end effector turned when not use while other tests are taking place. The high clock rate of the driver meant that it was generating a lot of heat just from being turned on without performing any function. The end effector team should have been more cognizant of this since we knew from the beginning that the driver would generate a lot of heat. Likewise, a good passive solution would have been to implement a heat sink to reduce the chances of burn out. However, with the help of Trudy Schwartz, we were able to find a couple compatible drivers with heat sinks built-in to serve the purpose of actuating the end effector pressure solenoid.

8.4 Sensors

When it comes to the sensor suite, the most problematic were the Pixy 2 sensors. A lot of lessons were learned when it comes to this sensor. The most important was to ensure that the environment has very bright and even lighting for the sensors to work optimally. This ensured the sides of the cargo didn't have shadows cast from the top of the box which would cause the hue of the colored markers to appear slightly different from the same colors on different sides of the box. The next lesson is that you pay for what you get. With the limited budget, the team was limited to very low budget cameras. The market tends to cater to low end or very high end equipment and doesn't have great selection in between the two categories. Anything better than the Pixy 2 with prepackaged recognition software capabilities would completely blow the budget. While on the low end, the Pixy 2 cameras could differentiate colored pixels rather nicely, but sometimes the camera software wouldn't give an output on the colored pixels it was seeing. This also couldn't be adjusted without modifying the source code of the cameras. Overall the software team felt having an in house color detection algorithm may have saved a lot of time and effort with the Pixy 2 cameras and would have significantly upped the performance of the system. If the Pixy 2 cameras were still utilized then the team also feels a sturdier mounting solution would have also been beneficial to prevent the constant need to re-calibrate the cameras every time the sensor cage was bumped or knocked. One last lesson would be to have a mounting solution that can be easily adjusted without hardware and hand tools. To adjust the cameras, screws had to be loosened and tightened consistently which caused some hardware to become stripped. Overall the mounting solution still worked as intended but caused many frustrations and wasted time throughout the use.

8.5 Software

This project consisted of a very complex and intricate software system. Developing this software taught us numerous lessons to take into our future work. One of the most significant lessons learned include to start software development and integration as early as possible. We were very rushed from a timeline perspective on the software team that starting early could have greatly helped mitigate. Next, before development begins, it is critical to plan individual chunks of software in as much detail as possible. Planning software is a difficult thing to do while it is in development, especially for such a complex project. Next, it is very important to clearly define all software-software and software-hardware interfaces before development begins. We ran into numerous issues while trying to integrate software with other software and while integrating software and hardware. Our interfaces were not defined clearly, which made it much more difficult to integrate the system overall and required significant amounts of time to resolve. Next, on software projects as large as this one, it is very important to create and maintain detailed documentation throughout the entire project lifecycle. This is especially important when multiple people are developing pieces of software in parallel that other people will have to use in the future.

More than anything, this software project taught us that the most important thing throughout a project is organization. Organization can make or break a software project, especially one as large and complex as this one.

8.6 Project Management

When it comes to managing such a large and complex project, it is essential that everyone that has taken a leadership role be flexible and communicative. One of the biggest reasons behind this team's success was the ability of all our team members to communicate delays, needs, and issues to everyone else on the team. During the course of the semester the PM and the lead systems engineer must be on the same page and hold each other accountable because these two positions are responsible for ensuring the project's various sub teams get the support and push necessary to finish on schedule. Always be conservative with planning for software development, debugging usually takes far longer than expected. In our experience a software bug can take up to 24 hours to debug. Furthermore, prior to presentations or major deliverables to the University it is very important to get advice from members of the PAB board in addition to the assigned advisor. The PM must lead by example, and should be in the thick of the engineering whenever possible while balancing the administrative needs of the team.

9 Individual Conclusions

Authors: All Team Members

Lindsay Cobb

Wrote sections 1, 2, 3.1, and 6. I worked primarily on requirement flow down and verification as well as testing plans and levels of success. I manufactured the 'cargo bags' used for testing. I also worked on the verification and validation of our requirements based on the testing plan and results.

Kyle Li

Wrote section 7 and parts of 3 and 8. During this semester I was focused on ensuring the team met all objectives, even the stretch objectives that we've planned out in the fall semester. Planning, enforcing team cohesion, and supporting various sub-teams during the course of the semester was of primary importance to me. I organized the majority, if not all, of the administrative tasks associated with scheduling, feedback, and communication with our SNC liaison and various professors. When not working on administrative tasks, I helped the end effector group and the software group with script writing and debugging.

Logan Vangyia Wrote the translator parts for section 3,4, and 8. Over the course of this semester I designed the manufactured parts for the translator and networked with other sub-teams with connections to the translator. I also aided the project manager with administrative tasks when needed as well as give a helping hand to other sub-teams when they were short staffed.

Marin Grgas Wrote the design description (section 3.2) for the end effector as well as the functional block diagram. During the semester I worked primarily on the end effector. I worked on integrating the various subsystems (end effector, solenoid, pressure regulation system) of the end effetor together. The solenoid was interfaced with the software system through the solenoid driver and Arduino. The sensors on the end effector were also integrated with the Arduino for software purposes. Furthermore, I performed many tests to characterize the end effector's capabilities. Finally, I wrote Arduino scripts for testing the functionality of the end effector, solenoid, and sensors.

Alex Ferguson Wrote the Cost Plan section (7.3). Over the course of this semester, I worked primarily as the finance lead for the team. I made orders on the team's behalf, tracked the packages, and coordinated the pickup from the Autoclave. I verified each item was successfully received. I also kept many spreadsheets up to date on our finances. This includes the expense tracking spreadsheet I am required to send to the financial representative for our senior project. Other sheets include a bill of materials, procurement status, big ticket items, and post-CDR purchases. I also aggregated the receipts and expenses and posted them to the university's Canvas system, ensuring the expected purchases did not exceed the budgetary limit allocated to to the team.

Jordan Abell Wrote the Estimation Algorithms section and the Bag Configuration Node section of the Software Manufacturing section. Wrote the Estimation Test section of Verification and Validation. Contributed to the System Test Results section. Wrote the software lessons learned section. Throughout the project, I was primarily involved in the development, testing, and integration of the software system, development and testing of the sensor system, integration and testing of the full system. I assisted in designing and constructing the test environment and end effector design and assembly.

Jett Moore Wrote the electrical parts of sections 3,4,6 and the translator segment of sections 5. Over the last semester I worked on getting the translator ready to mount, interfacing all of the electronic components, programming the stepper motor Arduino script and testing the translation system. In the last couple weeks of the semester I helped with combining the Arduino software, building a new solenoid driver and recording and editing the symposium video.

Nicholas Miller Wrote sections 3.2.3.1, 5.2.1, and 5.2.2. Throughout the past semester I worked on building and testing the end effector. Making sure that all of its components worked properly. I also assisted in creating the code that controlled the actuation of the end effector and the operation of the actuation and grab sensors, which are mounted on the end effector. Integrated the end effector subsystem into the overall system, working with its various pneumatic components.

James Tiver Wrote sections 3.2.1 ; 3.2.4; 3.2.5; 4.4; 5.4. Over the last semester I worked on optimizing and testing the Pixy 2. I worked on writing the Arduino code for the Pixy 2. I helped with building and assembling the sensor cage and mounting the cameras. I also worked on Arduino ROS communications and helped with integrating the ROS components. Finally towards the end of the semester I helped out with integrating and testing the entire system before the Symposium.

Bruce Barnstable Over the last semester, I planned and assisted in the construction of the PVC - sensor mounts. I also assisted in testing of the UR10 robotic arm with proprietary software. I also developed the CAD Camera Model for the CORE Environment.

Brandon Torczynski Wrote section 4. Planned and performed tests to ensure the functionality of the translator system. Worked on the fabrication and construction of the end brackets, baseplate, coupler, and stepper motor mount. I also mounted the pressure regulator, constructed mounts for the limit switches, soldered wires, and organized wire cables and pressure hoses to be adequately used.

Peter Amorese Wrote the manufacturing design for the Task Planner Node, Task Planning Library, Status Node, and Manipulator Node, as well as the ROS Environment Test section. This semester I was responsible for creating the Task Planning Library, as well as developing a large portion of the ROS environment. I helped with testing and integrating software and hardware components for full system functionality.

Bibliography

[1] K. He, M. Lahijanian, L. E. Kavraki and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 346-352, doi: 10.1109/ICRA.2015.7139022.

[2] https://www.arduino.cc/en/uploads/Main/arduino-Ethernet-Shield2-V2-sch.pdf

[3] "Optimal State Estimation" *Simon, Dan.* Cleveland State University. 2006.

[4] "Spot" Online LTL Toolset: https://spot.lrde.epita.fr/app/

Appendices

A Trade Studies

A.1 End Effector

A.1.1 Possible Solutions

The end effector is the mechanism by which the robotic arm will interface with the cargo bags. It is responsible for capturing and securing the cargo bag during translation. Six candidates were investigated in the end effector trade study: Velcro, a vacuum, a magnetic attachment, a gripper, a latching mechanism, and a gecko gripper. Each of the design solutions are discussed in more detail below. Images of each system can be found in appendix C.1.

A.1.1.1 Velcro

Velcro is the simplest solution to capture and secure cargo bags. It is lightweight, cheap, strong, and easily available. However, NASA has noted many issues with Velcro [**?**] learned through use on the Apollo program and the ISS. Table 8 lists the significant pros and cons of using Velcro as the end effector.

Description	Pro	Con
Lightweight	X	
Cheap	X	
Strong	X	
Environmental Hazard		X
Must be replaced when worn out		X
Additional hardware, software, and electronics required to achieve mission tasks		X

Table 8: Velcro Pros and Cons

A.1.1.2 Vacuum

The vacuum end effector under investigation for the trade study is the OnRobot electrical vacuum gripper, the VG10. For the vacuum gripper to work the cargo bags would require a non-porous side, such as hard plastic. The data for the VG10 was taken from the OnRobot VG10 datasheet [?]. Figure ?? shows the VG10. Table 9 lists the significant pros and cons of using the VG10 end effector.

Description	Pro	Con
Capable of exerting a large force	X	
Must have a non-porous surface to pick up		X
Large power requirement		X
Comparatively high weight		X

Table 9: Vacuum Gripper Pros and Cons

A.1.1.3 Magnetic

The magnetic end effector under investigation for the trade study is the SMC MHM-25. This end effector utilizes pneumatic pumps to control the position of a magnet. This gripper requires air pressures between 0.25 and 0.6 MPa. Each cargo bag will need a small sheet of magnetic material on one of the sides to allow the gripper to attach. The data for the SMC MHM-25 was taken from the SMC MHM-Series magnetic gripper datasheet [?], and the auto-switch sensors datasheet [?]. The magnetic gripper can be seen in Figure ??. Table 10 lists the significant pros and cons of using the magnetic end effector. Magnetic force obeys the inverse square law, $F \propto \frac{1}{r^2}$ and by manufacturer specifications, any material outside a radius of about 17 mm from the end effector will not be affected.

Description	Pro	Con
Capable of exerting a large force	X	
Must have a magnetic surface to hold		X
Additional components necessary for operation		X
Reusable	X	
Low power requirement	X	

Table 10: Magnetic Gripper Pros and Cons

A.1.1.4 Gripper

The next end effector under consideration is the OnRobot RG6 Gripper. The RG6 has a mass of 1.25 kg and a volume of $2.3 * 10^{-3} m^3$. It can grip with a force of 120 N and requires from 1.4 to 15 W of power. The data for the RG6 was taken from the OnRobot datasheet [?]. The OnRobot RG6 can be seen in Figure ??. Table 11 lists the significant pros and cons of using the RG6 end effector.

Description	Pro	Con
Capable of exerting a large force	X	
Large power requirement		X
Out-of-the-Box ready	X	
Reusable	X	
Versatile functionality and uses	X	

Table 11: Gripper Pros and Cons

A.1.1.5 Latch

The device under investigation for the latching mechanism end effector is the R4-EM-R712-131. Data for the R4-EM-R712-131 was taken from the performance data sheet [?]. Figure **??** and **??** show the R4-EM-R712-131 latching mechanism. Table 12 shows the pros and cons of the device for end effector use within the scope of this project. Significant modification is necessary in order to use this latching mechanism as an end effector for the robotic arm.

Description	Pro	Con
Capable of exerting a large force	X	
Requires significant modification to achieve mission tasks		X
Reusable	X	
Moderate power requirement	X	

Table 12: Latch Pros and Cons

A.1.1.6 Gecko Grip

The end effector under investigation for the Gecko Grip is the OnRobot Gecko SP5. This end effector uses an adhesive pad to grip surfaces. Successful operation of the Gecko Grip requires highly polished surfaces to "allow for maximum contact between the adhesive pads and the substrate surface." Over time of continued use the adhesive pads will wear out and require replacement. Data for the Gecko SP5 was taken from the OnRobot Gecko SP1/3/5 datasheet [?]. The Gecko Grip is shown in Figure **??**. Table 13 lists the significant pros and cons of using the Gecko Grip as the end effector.

Description	Pro	Con
Capable of exerting a small force		X
Must have highly polished, solid surface		X
No electronics or compressed air necessary for operation	X	
Max of 200 cycles before maintenance		X

Table 13: Gecko Gripper Pros and Cons

A.1.2 Trade Study

The evaluations of the end effector solutions were based on seven metrics: mass, volume, strength, out-of-the-box readiness, cargo bag modification, mating, and power requirements. The goal is to evaluate the possible design solutions using clear, objective metrics to avoid bias.

A.1.2.1 Mass of End Effector

Mass is a crucial metric in space environments due to constraints launch mass. Since the system is intended to be a proof-of-concept for a space robotic system, mass should still be considered but will have a low weight as a result. The scale for this metric was decided based on the order of magnitude of masses for the end effectors under consideration. The highest mass end effector was 1.25 kg, so the most logical scale was a linear scale from < 0.1 kg to ≥ 1 kg. Table 14 shows the scoring breakdown for the end effector mass evaluation metric.

Metric Weight: 10%

Score	5	4	3	2	1
Mass [kg]	$<$ 0.1	0.1 - 0.39	0.4 - 0.69	0.7 - 0.99	$\geq$ 1.0

Table 14: End Effector Mass Scoring Breakdown

A.1.2.2 Volume of End Effector

Driving Design Requirements: DR 9.1

Similarly to the mass of the system, volume is another important metric to consider for space systems since launch volume and operating volume are both limited. The end effector's volume is relatively small compared to the core and is thus assigned a low weight. The largest volume of the end effectors under investigation is 0.016 m^3. Therefore, the evaluation scale was linear from $<$ 0.001 to $\geq$ 0.01 m^3. Table 15 shows the scoring breakdown for the end effector volume evaluation metric.

Metric Weight: 10%

Score	5	4	3	2	1
Volume [m^3]	$<$ 0.001	0.001 - 0.0039	0.004 - 0.0069	0.007 - 0.0099	$\geq$ 0.01

Table 15: End Effector Volume Scoring Breakdown

A.1.2.3 Strength

Driving Design Requirements: DR 1.2

The strength of the end effector determines the force that the system can apply to transport cargo bags through the LIFE module. The scale for the strength metric was determined by the operating range of forces. The increment between scores was set to 1 N so that the max score of 5 was approximately equal to the maximum force that the UR-10 robotic arm can apply, which is approximately 100 N. Table 16 shows the scoring breakdown for the end effector strength evaluation metric.

Metric Weight: 20%

Score	5	4	3	2	1
Strength [N]	$>$ 96	90 - 95	85 - 90	80 - 85	$<$ 80

Table 16: End Effector Strength Scoring Breakdown

A.1.2.4 Out-of-the-Box Readiness

The out-of-the-box readiness design score is intended to evaluate the amount of work that will be required to develop a particular end effector for the system design. This category refers to three essential tasks of the end effector which are defined as: (1) Grab the cargo bag, (2) Secure the bag during translation, and (3) Release the bag. The best score corresponds to an end effector that is capable of completing all three essential tasks with no modifications necessary. Table 17 shows the scoring breakdown for the end effector out-of-the-box readiness evaluation metric.
Metric Weight: 15%

Score	5	4	3	2	1
Out-of-the-Box Readiness (Capable of all three essential tasks:)	without modification	with custom robotic arm mounting necessary	with modifications to one of the three: Hardware, Software, or Electronics	with modification to two of the three: Hardware, Software, or Electronics	with modification to all Hardware, Software, and Electronics

Table 17: End Effector Out-of-the-Box Readiness Scoring Breakdown

A.1.2.5 Bag Modification

Bag modification is a measure of how much mass needs to be added to the bag to be compatible with the end effector. The bag size was estimated using the standard half Cargo Transfer Bag (CTB), which is one of the standard bags used on the ISS. The scale for bag modification was set to start at 0 kg for a score of 5 (indicating no bag modifications), with 1 kg increments between scores, up to > 3 kg. Table 18 shows the scoring breakdown for the end effector bag modification evaluation metric.
Metric Weight: 15%

Score	5	4	3	2	1
Bag Modification [kg]	0	0 - 1	1 - 2	2 - 3	> 3

Table 18: End Effector Bag Modification Scoring Breakdown

A.1.2.6 Mating

The mating evaluation metric is intended to capture the ease of mating the end effector with a cargo bag. Evaluations are based on DOFs needed to accomplish mating. 0 DOF means that the end effector does not need precise positioning or orientation to mate with a bag. Table 19 shows the scoring breakdown for the end effector mating evaluation metric.
Metric Weight: 10%

Score	5	4	3	2	1
Mating	0 DOF	—	3 DOF	—	6 DOF

Table 19: End Effector Mating Scoring Breakdown

A.1.2.7 Power Requirements

Driving Design Requirements: DR 2.1
The final evaluation metric for the end effector is its power requirements. While power is important on flight-ready models, our proof-of-concept design has no such limitations and power is this assigned a low weight. The power draw of the end effectors under investigation is on the order of 0 - 10 W, so the scoring scale for the metric was set linearly between < 1 W and ≥ 10 W. Table 20 shows the scoring breakdown for the end effector mating evaluation metric.
Metric Weight: 20%

Score	5	4	3	2	1
Power [W]	< 1.0	1.0 - 3.9	4.0 - 6.9	7.0 - 9.9	≥ 10.0

Table 20: End Effector Power Scoring Breakdown

A.1.2.8 Trade Study Results

The results of the end effector trade study are shown below in Figure 52. The scores of the trade study indicate that the top three design choices for the end effector are (1) the Magnetic gripper end effector, (2) Velcro, and (3) the Gripper.

End Effector							
Metric	**Weight**	**Velcro**	**Vacuum (VG10)**	**Magnetic (MHM-25)**	**Gripper (RG5)**	**Latch (R4-EM)**	**Gecko Grip (SP5)**
Mass [kg]	10.0%	5	1	4	4	5	4
Volume [m^3]	10.0%	5	1	5	4	5	5
Strength [N]	20.0%	5	5	5	5	5	1
Out-of-the-Box Readiness	15.0%	1	5	2	5	1	4
Bag Modification [kg]	15.0%	5	4	4	4	3	3
Mating	10.0%	3	5	5	3	1	3
Power [W]	20.0%	5	1	5	1	3	5
Total	100.0%	4.2	3.25	4.3	3.65	3.3	3.45

Figure 52: End Effector Trade Study Results

A.1.3 Selection of Baseline Design

After evaluating both the results of the end effector trade study and the pros and cons of the top three designs we have decided that our baseline design will include the SMC MHM-25 Magnetic Gripper. The MHM-25 scores slightly higher than Velcro, but the drawbacks of Velcro are significant enough to remove it from consideration. Additionally, the MHM-25 scores sufficiently higher than the RG6 to justify selecting the MHM-25 for our baseline design. This design option provides the best solution to our design problems, with the most minor drawbacks.

A.2 Sensor Suite for Bag Configuration

In order for the IVR system to successfully unload docked cargo, a type of sensor system will need to be implemented for identifying the position and configuration of the cargo bags being unloaded. Potential options to accomplish this task include lidar sensors, sonic rangers, and image processing. Data processing from these sensors will be a combination of automated software and user input.

A.2.1 Design Solutions

A.2.1.1 Lidar Sensor

Lidar (light detection and ranging) is a method of measuring distances by pulsing a laser light at an object and leveraging measurements of the reflected light to assemble 3-D representations of the package. The return measurements include measuring the time needed for the laser to return and the associated wavelength of the reflected light. Lidar sensors have high detail and can map physical features with very high resolutions. A picture of a lidar system can be seen in figure **??** in the appendix as well as a pros and cons below in table 21.

Description	Pro	Con
Wide range of application (long distances, short distances, high or low detail)	X	
Highly accurate	X	
Poor operation in high intensity light		X

Table 21: Lidar Pros and Cons

A.2.1.2 Ultrasonic Ranger

This type of sensor operates by utilizing a transducer to emit a high frequency sound (higher than humans can hear) towards an object and measures the sound reflected back. The sensor can emit the sound by receiving a digital pulse from a micro-controller connected to it and can detect objects by measuring the return time of the signal and comparing to the expected return time. In the appendix a diagram of an ultra sonic ranger can be seen in figure **??** as well as a table summarizing pros and cons below in table 22

Description	Pro	Con
Generally low power consumption	X	
Environmental versatility	X	
Object dependent (if bag is wool it might absorb the sound)		X
Shorter object detection range		X

Table 22: Sonic Ranger Pros and Cons

A.2.1.3 Image Processing

Image processing can be the act of distinguishing what is the object of interest from the surroundings. it can also incorporate any other calculations associated with the object of interest such as range or location. The majority of these tasks is heavily software dependent. There are multiple third party libraries that can be utilized for either microprocessors or full sized computers respectively, however they may be difficult to understand and implement. Image processing "kit" refers to the out-of-the box complete kit with cameras equipped for motion and configuration determination. This includes the software and data processing. The group will search for something feasibly in the budget between these two extremes. An example of a camera which will need more user programming can be seen in figure **??** in the appendix. A pros and cons can also be seen in table 23

Description	Pro	Con
Generally inexpensive	X	
High detailed mapping	X	
Position and orientation	X	
Computationally expensive		X
Long processing times		X

Table 23: Image Processing Pros and Cons

A.2.2 Sensor Suite Trade Study

The Sensor Suite trade study was evaluated based on DR 3.1, DR 3.2, DR 3.3, and DR 3.4. The metrics selected for this trade study are: software complexity, accuracy, range, mounting capability and field of view, ability to determine bag versus environment, and power requirements. This section will explain the rationale for including each metric in the analysis, along with the weight of each category and the evaluation scale.

A.2.2.1 Software Complexity

Driving Design Requirements: DR 3.1, DR 3.2, DR 3.3, DR 3.4
Software complexity was included to evaluate the scope and efficacy of the software needed to take the sensory information and convert it into a position and orientation measurement. While complexity is a factor in time investment, it was determined that the quality of the sensor suite should be weighted more heavily as good data decreases the need for software complexity to process data.
Metric Weight: 10%

Score	5	4	3	2	1
Software Complexity	Out of the box functionality and localization	NA	NA	NA	Coding functionality from scratch using only sensory information

Table 24: Software Scoring

A.2.2.2 Accuracy

Driving Design Requirements: DR 3.1, DR 3.2
Accuracy refers to the accuracy of the sensed position and orientation relative to the true position and orientation. The lower bound determines the position within 3cm but knows nothing about the orientation. The upper bound can detect the position within .3cm and orientation within 3 degrees. This allows for more reliable pick up in a high cost environment. This is weighted more heavily because the mission can't be completed if the bag can't be found accurately enough.
Metric Weight: 20%

Score	5	4	3	2	1
Accuracy	Location of objects determined to within .3cm and 3 degrees	2.1 cm 17°	1.2 cm 31°	3.0 cm 45°	3cm no orientation

Table 25: Sensor Accuracy Scoring

A.2.2.3 Range

Driving Design Requirements: DR 3.1, DR 3.2, DR 3.3
The range metric was included to account for the sensor suite's ability to span the entire core. The lower bound was made assuming a sensor could be in the hatch doorway with the package. This distance would be the smallest possible range a sensor could utilize in the LIFE module. This metric allows the group to decipher how many sensors are needed and what all can be seen in the core. This also helps with determining if the cargo was transported successfully or not. This category was weighted more on the moderate side since low scores could be offset with more sensors.
Metric Weight: 15%

Score	5	4	3	2	1
Range	∞	5.6m	3.9m	2.3m	0.63m

Table 26: Sensor Range Scoring

A.2.2.4 Mounting Capability and FOV

Driving Design Requirements: DR 3.1, DR 3.2, DR 3.3, DR 3.4
Mounting Capability and Field of View refers to the ability for the entire sensor suite to capture the spaces that need to be analyzed. Mounting capability refers to the possible positions that a single (or subgroup of) sensor(s) can be mounted such that the field of view is not obstructed. Mounting Capability and FOV is complimentary to the "Range" metric. The consequences for a low score on this metric are similar to that of "Range" in that increasing the amount of sensor groups can solve the short-comings.
Metric Weight: 15%

Score	5	4	3	2	1
Mounting Capability and FOV	Full spherical FOV, not dependent upon mounting location	Can be mounted in most places within the core while maintaining full necessary FOV (1 sensor)	2 sensor groups are needed	4 sensor groups are needed	Mounting location heavily restricted by the FOV, more sensors are needed for full localization.

Table 27: Sensor FOV and Mounting Scoring

A.2.2.5 Ability to Distinguish Bag

Driving Design Requirements: DR 3.1, DR 3.2, DR 3.3, DR 3.4
The ability to distinguish what is a bag and what is only part of the environment is incredibly important when considering sensor suites. The sensory data that is being output from these sensor suites must be able to be used to determine where a bag or piece of cargo is. This is how completion or irreversible failure is defined with respect a given task.This metric was heavily weighted to emphasize the importance of object differentiation between the environment. If a solution scores low on this metric, the sensory information provided by this sensor system alone can not locate the bag.
Metric Weight: 20%

Score	5	4	3	2	1
Determining Bag vs Environment	Sensor only recognizes bag	NA	NA	NA	Additional hardware or software needed to differentiate bag from the environment

Table 28: Determining Bag vs Environment

A.2.2.6 Power Requirements

Driving Design Requirements:
The sensors will utilize the smallest portion of the available power budget, but it's still crucial to budget and monitor because space systems have very limited power draw. This design however, is for a ground proof of concept test, but the group is still aiming to make it realistic to a flight design. The motivation for this is to minimize the obstacles when transitioning from a proof of concept to a flight ready system. **Metric Weight: 20%**

Score	5	4	3	2	1
Power [W]	0 - 10	10 - 20	20 - 30	30 - 40	$>$ 50

Table 29: Sensor Power Scoring

A.2.2.7 Sensor Suite Trade Study Results

Below are the results of all the sensor suite options being evaluated based on the scoring criteria. As can be seen by the results, the two best options are either image processing or an image processing kit with user friendly interfacing and software.

Sensor for Bag Configuration					
Metric	**Weight**	**Lidar Sensor**	**Sonic Ranger**	**Image Processing Kit**	**Image Processing**
Software complexity	10.0%	1	1	5	1
Accuracy	20.0%	1	1	5	5
Range	15.0%	5	4	4	4
Mounting capability and FOV	15.0%	3	3	3	3
Determining Bag vs Environment	20%	1	1	5	5
Power Requirements	20.0%	5	5	1	1
Total	100.0%	2.7	2.55	3.75	3.35

Figure 53: Sensor Suite Trade Study Results.

A.2.3 Sensor Suite Baseline Design

The results of the trade study found the two best options to be an image processing kit with user friendly interfacing and software followed by a stand-alone image processor. The only real difference between these two is the software. The package decided on was the Pixy 2 camera. It is an RGB camera with a built in processor that can be used to identify specific objects based on the hue of their color. The main advantage to utilizing this camera is to prevent the need for excessive programming of object detection. This helps keep the scope of the overall project in check for two semesters of work. To see an image of the down selected design please see figure **??** in the appendix.

A.3 Cargo Bag

A.3.1 Baseline Design

The cargo bag needs to be identified, retrieved and delivered from a starting location to a drop off location. The baseline design of the cargo bag was driven by end effector and sensor suite choices while taking influence from NASA Cargo Transfer Bag (CTB) that is already in use. The baseline design includes only one transfer bag modelled after NASA's half-size CTB and a single bag attachment. The material of the baseline design will begin with cardboard and increase in fidelity for future iterations.

Table 30: NASA Cargo Transfer Bag

CTB Size	Internal Dimensions (cm)	Mass of CTB (kg)	Max Cargo Mass (kg)
Full Size	41.27 x 24.13 x 49.66	1.68	25.54
Half Size	41.27 x 22.86 x 24.13	0.84	12.63

The Bag Attachment is driven by the end effector choice. For a magnetic end effector, the attachment on the cargo bag will a steel plate with a 59 millimeter diameter and 6 millimeter width. This allows a 5 millimeter margin for the magnetic end effector to interface with the steel plate surface. The Bag Identification Tags are driven by the sensor suite choice. For the Pixy2 Camera object recognition program, objects can be identified by either identification codes or color codes. In order to give identifiable tags on 5 different locations for each face of the cargo bag, the baseline design will use color codes due to the wider variety of identifier options.

A.4 Linear Transport System

The linear transport system has the task of moving the cargo along with other relevant hardware anywhere along the core within the life module. This translator need to anchor the entire system to the life module walls as well as handle any forces or moments created during the movement of the cargo.

A.4.1 Possible Design Solutions

A.4.1.1 Ball Bearing with Conveyor Belt

This system moves the robotic arm by spinning a flexible belt. Ball bearings instead of pulleys to hold the belt has the advantage of reducing the belt's frictional degradation, which improves the durability of the system and reduces the amount of maintenance needed. The concept of the conveyor design is depicted in figure **??** and **??** in the appendix. However, mounting the robotic arm to the LIFE module is complicated and delicate. A potential way to mitigate high torques on the robotic arm would be to lower the translational velocity.

Description	Pro	Con
Complex		X
Highly accurate	X	
Low Power Cost	X	
Fast movement		X

Table 31: Conveyor with Ball Bearing Pros and Cons

A.4.1.2 Wheels with Motor

The wheels with a motor design is a simple method similar to roller coasters. Using at least a dual set of wheels above and below the upper edge of the I-beam, the design could hold the translating platform to the wall of the LIFE Module without floating away shown in Figure **??** in the appendix. A potential problem with this design in zero gravity is that a wheel needs a normal force in order to roll the system forward. This could cause a major increase in fiction.

Description	Pro	Con
Simple	X	
Highly accurate	X	
Potentially high power cost		X
Fast movement		X

Table 32: Wheels with Motor Pros and Cons

A.4.1.3 Ball Bearing with Threaded Rod

The threaded rod design uses guide rails with ball bearings that are attached to the platform shown in Figure **??** in the appendix. A motor powers rotates the threaded rod attached to the platform that will move the platform. This design has the benefit of being reliable in both 1G and microgravity. The downsides to this design are its slow translation speed and need for extra materials to install it on the LIFE Module.

Description	Pro	Con
Simple	X	
Highly accurate	X	
Extra Materials		X
Slow movement	X	

Table 33: Ball Bearing with Threaded Rod Pros and Cons

A.4.1.4 Rack and Pinion

This device drives the motion along a track shown in Figure **??** and **??**. While extremely durable and capable of handling high payloads, extensive use makes this device imprecise overtime as the gears become worn. In addition, the movement of this slide needs to be programmed and calibrated to work in tandem with the robotic arm.

Description	Pro	Con
Complex		X
Weight		X
Carrying Capacity	X	
Precision	X	

Table 34: Rack and Pinion

A.4.1.5 Magnetic Levitation

The propulsion behind magnetic levitation is a linear induction motor. This method provides a high ceiling of force as well as velocity. Drawbacks to the linear induction motor is high weight and power consumption.

Description	Pro	Con
Force	X	
Speed	X	
Power Consumption		X
Weight		X

Table 35: Magnetic Levitation Pros and Cons

A.4.1.6 Single Axis Linear Stepper Motor

The stepper motor is guided by linear roller bearings along a platen provided by the manufacturer. Unlike a traditional motor, a linear motor provides a direct drive instead of torque through a non-contact method. This reduces wear and tear and provides a higher degree of accuracy.

Description	Pro	Con
Lightweight	X	
Low Power	X	
Precise	X	
Open Loop Control		X

Table 36: Single Axis Linear Stepper Pros and Cons /cite

A.4.2 Linear Transport System Trade Study

The linear transport system trade study was constructed by combining the propulsion and means of translation for the system. These components couldn't be evaluated separately due to their co-dependency. The metrics selected for this trade study are listed below.

A.4.2.1 Mass of Translator

Driving Design Requirements: DR 1.1, DR 3.1

The translator mass metric is specific to section of the system that will be moving with the robotic arm. This does not include the rails, or other stationary aspects of the system. The motivation for this metric is the need for the system to be tested in 1G. The lower the mass of the translator, the easier it will be to accelerate.
Metric Weight: 10%

Score	5	4	3	2	1
Translator Mass[kg]	0-2	2-4	4-6	6-8	>8

A.4.2.2 Supporting Force

Driving Design Requirements: DR 3.1
Supporting force is a metric to ensure that the translation system can support the forces created by the robotic arm. This metric is driven by the requirement for the system to be validated in a 1G environment. The force was calculated using a mass of 33.5kg for the arm and 10 kg for the maximum payload. This totals to 435 N using newtons second law.
Metric Weight: 10%

Score	5	4	3	2	1
Supporting Force(N)	>500	499-484	483-468	467-451	450-435

A.4.2.3 Translation Force

Driving Design Requirements: DR 3.1
Translation force determines the system's acceleration while translating along the length of the LIFE module's core. A higher force enables greater acceleration. This metric was given a weight of 10% to reflect its importance in demonstrations for a 1G environment.

Metric Weight: 10%

Score	5	4	3	2	1
Translation Force [N]	>30 N	25-30N	20-25 N	15-20 N	<15N

A.4.2.4 Repeat-Ability

The repeat-ability metric is included to ensure that the system can run off the resources provided in the LIFE module with little to no maintenance as well as taking into consideration the extra resources needed for each design. As such the repeat-ability metric is included to reduce complexity in the system and ensure the design has high longevity with the given resources.
Metric Weight: 15%

A.4.2.5 Precision

Driving Design Requirements: DR 5.2
The precision metric ensures that the translation system is able to fulfill DR 5.2. The design requirement only requires a precision of 5 cm. However, especially for systems with open loop localization it is essential that the system is able to move precisely enough for the process to be automated for multiple cycles. To account for this the scoring for precision was constructed between 0-7mm.
Metric Weight: 20%

Score	5	4	3	2	1
Precision [mm]	<1	1-3	3-5	5-7	>7

A.4.2.6 Localization

Driving Design Requirements: DR 5.1 The localization metric is based on how complicated the methods are for the system to determine it's position along the track. If the system is able to determine it's position out of the box it is given a 5, while if the system will need an additional sensor suite, it is given a 1. This metric is included in the trade study in an attempt to minimize the hardware and software complexity of the required position tracking system.
Metric Weight: 15%

Scoring	5	3	1
Localization	System includes closed loop localization	Precise open loop control localization resources to repeat	External sensor suite required for localization

A.4.2.7 Power Requirement

Driving Design Requirements: DR 2.2, DR 2.3
The power requirement metric is important to include as any space-borne mission will involve power limitations. The high end of scoring is set at 0 W because less power consumption will allow more margin for power surges and more power allotment for other sections of the IVR system.
Metric Weight: 20%

Score	5	4	3	2	1
Input Power[W]	0-60	60-120	180-240	240-300	>300

A.4.2.8 Linear Transport System Trade Study Results

Below are the results of all the Translation system options being evaluated based on the criteria previously examined. Certain qualities are weighted more heavily than others based on Functional requirements given to us by Sierra Nevada Corporation.

Linear Transportation System							
Metric	Weight	Maglev	Conveyor	Rack and Pinion	Threaded Rod	Motorized Wheels	Single Axis Stepper
Mass of Translator [kg]	10.00%	4	2	3	4	3	5
Supporting Force [N, Nm]	10.0%	5	5	4	5	3	5
Translation Force [N]	10.0%	5	3	4	5	3	5
Reusibility	15.0%	5	3	3	3	3	5
Precision [mm]	15.0%	4	4	4	5	5	5
Localization	20.0%	3	3	3	5	5	3
Power Requirements [W]	20.0%	1	4	3	2	3	5
Total	100.0%	3.55	3.45	3.35	4	3.7	4.6

Figure 54: Translator Trade Study Results

A.4.3 Translator Baseline Design

The design that was selected from the trade study was the ball bearing with the threaded rod since it could best satisfy the requirements without exceeding the budget of the project like the single axis stepper. A CAD model of the translator system is pictured in Figure **??** in the appendix

Mass	60kg
Dimensions	Length:2 meters Width: 30cm Height: 9cm
Power	24V/3A/72W

Table 37: Basic Characteristics of Translator system

The system is powered by a NEMA 23 stepper motor that rotates a threaded rod that runs through a lead nut attached to the underside of the baseplate causing the platform to move back and forth when the threaded rod is rotated. The robotic arm will be bolted to the top of the baseplate. This baseplate then sits atop two linear guide rails that allow for low friction 1-dimensional movement. The motor and end brackets hold the different components together while also providing a easy way to clamp the entire system to a table such that it will not move. The baseplate and brackets will be manufactured by our team while the remaining will be purchased and incorporated into the overall design.