DESIGN AND IMPLEMENTATION OF AN ADAPTIVE ROBOT GRIPPER

By

JOE LIANG

A Thesis Submitted to The W.A. Franke Honors College
In Partial Fulfillment of the Bachelors degree
With Honors in
Electrical and Computer Engineering
THE UNIVERSITY OF ARIZONA
M A Y  2 0 2 2

Approved by:

____________________________
Dr. Larry Head
Department of Electrical and Computer Engineering
Abstract

The Adaptive Robot Gripper (ARG) project has the purpose of designing and producing a prototype of a universal gripper for Unilever production lines. This device is meant to have extended capability beyond their current industrial robot grippers and will provide the ability to pick up products of various shapes, sizes, and weights as well as have capability to pick up products with differing package rigidity without damage to the packaging.

The ARG will be comprised of a camera system, a gripper, and emulation software. The camera system will identify the location of the product while the size and shape of the recognized item will be retrieved from stored data in the Raspberry Pi. This information will be sent to the arm and gripper so that the product will be handled carefully. The included emulation software will simulate the industrial robot arm and gripper in the manufacturing environment. The virtual simulation of the arm will show how the robot arm would respond to the location of products on a conveyer belt and show how it would orientate itself to pick up and drop each Unilever Product.

This adaptability of the system will be used in the manufacturing line to increase efficiency as the lines no longer will need to stop for grip changes. This final report walks through all the aspects of project, documents all subsystems, and clearly states the data from testing.
Team Member Roles and Responsibilities

Lucas Hawley – Team Lead
Lucas Hawley is a senior majoring in Biomedical Engineering and was the team lead of the adaptive robot gripper project. He oversaw setting up meetings with the sponsors, sending in project status reports, and keeping track of all team member and project progress. All reports, presentations, and other documents were sent to Lucas for final submission. Lucas was also in charge of creating the Design Day video and the poster board, as well as the graphical user interface (GUI) for the adaptive robot gripper.

Connor Nagore – Procurement Lead
Connor Nagore is a senior majoring in Mechanical Engineering and was the procurement lead of the adaptive robot gripper project. He oversaw all purchases of the project, sent in order forms, and tracked the budget and links. Connor was also instrumental in the RoboDK emulation software development and wiring diagrams.

Riyadh Alswayel – Mechanical Lead
Riyadh Alswayel is a senior majoring in Mechanical Engineering and was the mechanical lead of the adaptive robot gripper project. He helped design, refine, and prototype the gripper of the project, as well as the interface plate which connects the gripper to a robot arm and the handle for design day demonstrations. Riyadh also created both plastic and aluminum prints of the gripper.

Max Critchfield – Optical Lead
Max Critchfield is a senior majoring in Optical Engineering and was the optical lead of the adaptive robot gripper project. Max helped create, redefine, and verify all the system requirements of this project, creating test sheets for all testing requirements as well as maintaining test data. He also aided in defining the proper step size and step count of the stepper motor from rigorous testing.

Joe Liang – Software Lead
Joe Liang is a senior majoring in Electrical and Computer Engineering and was the software lead of the adaptive robot gripper project. He helped write, debug, and integrate all the software components of this project, including the GUI, distance sensor, camera/image recognition, force sensor, stepper motor, and emergency stop button code. Joe was also vital in figuring out how to deploy all this existing software onto a Raspberry Pi, downloading all the needed libraries such as OpenCV.
Adaptive Robot Gripper

Final Report

Version: 1

Team: Lucas Hawley, Connor Nagore, Joe Liang, Riyadh Alswayel, Max Critchfield

Sponsor: Unilever
# Table of Contents

Table of Contents ................................................................................................................................. 5  
Scope and Introduction ............................................................................................................................. 6  
  Functional requirements ....................................................................................................................... 6  
System description / System block diagram ......................................................................................... 7  
  Statement of requirements and verification Requirements: .............................................................. 8  
  Verification Matrix and Results: .......................................................................................................... 9  
Models/ Analyses summary .................................................................................................................... 10  
TDP summary ....................................................................................................................................... 21  
Acceptance Test Results ....................................................................................................................... 22  
Final Budget .......................................................................................................................................... 33  
Lessons Learned ................................................................................................................................... 37  
Next Steps ............................................................................................................................................ 42  
Summary ............................................................................................................................................... 46  
Appendix ............................................................................................................................................... 47  
RoboDK User Manual ............................................................................................................................. 55  
Technical Data Package .......................................................................................................................... 101  
Design Documentation IDL: .................................................................................................................. 126  
Software Design Document: .................................................................................................................. 142  
Maintenance .......................................................................................................................................... 193  
Extra Credit: Engineering Standards Used in the Adaptive Robot Gripper Project ......................... 191
Scope and Introduction

The Adaptive Robot Gripper (ARG) project has the purpose of designing and producing a prototype of a universal gripper for Unilever production lines. This device is meant to have extended capability beyond their current industrial robot grippers and will provide the ability to pick up products of various shapes, sizes, and weights as well as have capability to pick up products with differing package rigidity without damage to the packaging.

The ARG will be comprised of a camera system, a gripper, and emulation software. The camera system will identify the location of the product while the size and shape of the recognized item will be retrieved from stored data in the Raspberry Pi. This information will be sent to the arm and gripper so that the product will be handled carefully. The included emulation software will simulate the industrial robot arm and gripper in the manufacturing environment. The virtual simulation of the arm will show how the robot arm would respond to the location of products on a conveyer belt and show how it would orientate itself to pick up and drop each Unilever Product.

This adaptability of the system will be used in the manufacturing line to increase efficiency as the lines no longer will need to stop for grip changes. This final report walks through all the aspects of project, documents all subsystems, and clearly states the data from testing.

The name SARAH was given to the prototypes of this project. This being an acronym for Simulated & Adaptable Robotic Attachment Hand. This not only gives our gripper prototypes a nice name for identification, but it also describes the purpose for this project.

Functional requirements

1. The ARG shall be used in an industrial/assembly line environment.
2. The ARG shall grab and release objects when appropriate.
3. The ARG shall be able to pick up items of different: sizes, shapes, densities, and weights.
4. The ARG shall connect/interface with existing robot arms.
5. The ARG shall function without a human operator.
System description / System block diagram

System Block Diagram

Team Responsibilities

Outlet/Power Supply
- 120 V (≈90,000 mW/h battery)
- Micro USB to USB
- OS: Wi-Fi

Display
- displays GUI of object recognition and sensor data
- Displays robot's movement

Raspberry Pi 4
- Object detection and recognition software
- Motor driver and force sensor software

Motor Driver
- Controls opening and closing of mechanical gripper

Grip Plate and Grip Material
- Provides friction
- Includes Teflon, Force Sensors
- Slippery, Force feedback

Mechanical Gripper System
- 2 Gripers
- 2 Grip Plates
- 2 Grip Links
- 1 Gripper Palm
- 2 Grips

Interface Plate
- Connects mechanical gripper to robot arm

Flexi-connector circuit (FPC)

Camera Stand
- Stationary camera stand

Camera System
- Aligned to center of robot

Object starts in visualization plane on robot/conveyor system

Object placed by Mechanical Gripper

Mat/Conveyor System
- Loading Zone
- Object stops at pickup location

Drop zone(s)
Statement of requirements and verification Requirements:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2.1 Camera System</strong></td>
<td><strong>2.1.1 Object recognition:</strong> Camera system shall detect and recognize the type of object to be gripped (95% accuracy)</td>
</tr>
<tr>
<td><strong>2.1.2 Stationary:</strong> The camera system shall be stationary next to the conveyer system &amp; the ARG shall have a default location</td>
<td></td>
</tr>
<tr>
<td><strong>2.2 Sensors and software</strong></td>
<td><strong>2.2.1 Data lookup:</strong> Pi shall retrieve all relevant data for each object</td>
</tr>
<tr>
<td><strong>2.2.2 Distance Detection:</strong> The ultrasonic sensor shall report object distances. (+/- 5mm)</td>
<td></td>
</tr>
<tr>
<td><strong>2.2.3 Force Sensor:</strong> The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips</td>
<td></td>
</tr>
<tr>
<td><strong>2.2.4 Emulation:</strong> The ARG shall utilize emulation software in place of a robot arm for demonstration purposes</td>
<td></td>
</tr>
<tr>
<td><strong>2.3 Performance and cost</strong></td>
<td><strong>2.3.1 Inexpensive:</strong> The cost of the ARG shall not exceed $4000</td>
</tr>
<tr>
<td><strong>2.3.2 Performance:</strong> The ARG shall have a success rate greater than 99%.</td>
<td></td>
</tr>
<tr>
<td><strong>2.3.3 Pick rate:</strong> The ARG shall have a pick rate of at least 30 picks per minute.</td>
<td></td>
</tr>
<tr>
<td><strong>2.4 Mechanical and Design</strong></td>
<td><strong>2.4.1 Adaptable:</strong> The ARG shall be able to grip all required Unilever products</td>
</tr>
<tr>
<td><strong>2.4.2 Handle:</strong> The ARG will have a handle for demonstration</td>
<td></td>
</tr>
<tr>
<td><strong>2.4.3 Safety:</strong> The ARG shall have a shutdown mechanism for emergency situations</td>
<td></td>
</tr>
</tbody>
</table>
### Verification Matrix and Results:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Verif Method</th>
<th>System/Subsystem Requirement</th>
<th>Measured/Predicted Value</th>
<th>References</th>
<th>Margin</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[Image 31x412 to 582x688]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Image 31x177 to 585x411]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Camera System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolos neural network model (95% accuracy)</td>
<td>D</td>
<td>1m DOF</td>
<td>&gt;1m</td>
<td>107100</td>
<td>N/A</td>
<td>Pass</td>
<td>The camera must have appropriate depth of field for object range.</td>
</tr>
<tr>
<td>2.1.2 Stationary: The camera system shall be stationary next to the conveyor system &amp; the ARG shall have a default location</td>
<td>I</td>
<td>Mount</td>
<td>1 meter width</td>
<td>N/A</td>
<td></td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.2 Sensors and software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2.1 Data lookup: PI shall retrieve all relevant data for each object</td>
<td>D</td>
<td>Data lookup</td>
<td>5 of 5</td>
<td>102600</td>
<td>N/A</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.2.2 Distance Detection: The ultrasonic sensor shall report object distances (± 3mm)</td>
<td>T</td>
<td>±0.03mm</td>
<td>±0.04mm</td>
<td>100121</td>
<td>5%</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.2.3 Force sensor: The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips</td>
<td>D</td>
<td>Force feedback</td>
<td>±1.71</td>
<td>107200</td>
<td>7%</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.2.4 Emulation: The ARG shall utilize emulation software in place of a robot arm for demonstration purposes</td>
<td>D/A</td>
<td>Synchronized</td>
<td>rtd</td>
<td>109000</td>
<td>N/A</td>
<td>Pass</td>
<td>Emulation will produce code to control the robot arm as shown in the simulation created.</td>
</tr>
<tr>
<td>2.3 Performance and cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000</td>
<td>A</td>
<td>&lt;$4000</td>
<td>$3270.20</td>
<td>$3269.80</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2 Performance: The ARG shall have a success rate of 95%</td>
<td>A</td>
<td>95%</td>
<td>95%</td>
<td>95%</td>
<td>1%</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.3.3 Pickup rate: The ARG shall have a pick rate of at least 30 picks per minute. (0.6 seconds/pick)</td>
<td>T/A</td>
<td>30 ppm</td>
<td>2s/pick</td>
<td>107400</td>
<td>.1 seconds or 16%</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.4 Mechanical and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products</td>
<td>D</td>
<td>5 unique products</td>
<td>5 of 5</td>
<td>107501</td>
<td>N/A</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.4.2 Handle: The ARG will have a handle for demonstration</td>
<td>D</td>
<td>Handle attachment</td>
<td>T&amp;D</td>
<td>103101</td>
<td>N/A</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2.4.3 Safety: The ARG shall have a shut-down mechanism for emergency situations</td>
<td>D</td>
<td>&gt;2 s</td>
<td>N/A</td>
<td>103102</td>
<td>N/A</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>
Models/ Analyses summary

Depth of field Demonstration:

Requirement and Justification:

2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolov5 neural network model (95% accuracy)

In order to operate in an industrial environment, the camera must be able to identify an object across the entire width of the conveyor transporting it. A 1-meter depth of field will ensure this capability. Depth of field is the distance between the nearest and the farthest objects that are in acceptably sharp focus in an image. In order to pass this demonstration, the camera system must correctly identify the object with the same 70% confidence from the object identification test across a 1m length without changing the camera’s focus.

Method and Data:

We tested each object at 3 points: one at the closest point where the objects are in the field of view (z=0m), another .5 meters away from the first point (z=.5m), and finally a point 1 meter away from the first (z=1m). The results are below:

<table>
<thead>
<tr>
<th>z</th>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0m</td>
<td>Tea – 0.800</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.847</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.752</td>
</tr>
<tr>
<td></td>
<td>Soap 0.786</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.930</td>
</tr>
<tr>
<td>.5m</td>
<td>Tea – 0.941</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.917</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.906</td>
</tr>
<tr>
<td></td>
<td>Soap 0.888</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.845</td>
</tr>
<tr>
<td>1m</td>
<td>Tea – 0.928</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.915</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.829</td>
</tr>
<tr>
<td></td>
<td>Soap 0.903</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.837</td>
</tr>
</tbody>
</table>

All values are greater than 70% or .7 so this demonstration was successful.
Stationary Inspection:

Requirement:

2.1.2 Stationary: The camera system shall be stationary next to the conveyer system & the ARG shall have a default location

Justification and Result:

In order to get consistent results and data, the camera system and ultrasonic sensor must be mounted statically. The image below shows the mount we made to achieve this goal.

Despite being made of LEGO parts the above mount is a cheap and sturdy housing for these two components of our system and satisfies requirement 2.1.2.
Data Lookup Demonstration:

Requirement:

2.2.1 Data lookup: Pi shall retrieve all relevant data for each object

Justification and Result:

Following a successful object identification, the software must be able to access the appropriate preprogrammed data from its code. The image below shows the relevant section of code which we have tested thoroughly.

Following the object identification, the respective set of commands will be followed.
**Emulation Demonstration:**

**Requirement and Justification:**

2.2.4 Emulation: the ARG shall utilize emulation software in place of a robot arm for demonstration purposes

Since a robot arm could not be sourced for this project, we will be manually positioning the gripper with a handle we’ve affixed to it. To demonstrate how the gripper will work in an industrial environment, we will show an emulation of our gripper on a robot arm operating on a conveyor belt. A screenshot is attached below.
Cost Analysis:

Requirement and Justification:

2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000

All senior design teams are given the same $4000 budget they must stay within. This makes budgeting and planning extremely important to ensure the project can be completed. A full breakdown of the budget for the adaptive robot gripper is shown below. There is also a second figure showing the difference between the total budget which includes extra parts, prototypes, and other development costs as well as the system cost.

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 8GB RAM</td>
<td>Labists</td>
<td>2</td>
<td>302.98</td>
</tr>
<tr>
<td>Camera Flex Cable</td>
<td>Amazon</td>
<td>1</td>
<td>8.79</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Amazon</td>
<td>1</td>
<td>12.99</td>
</tr>
<tr>
<td>Buttons 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>6.39</td>
</tr>
<tr>
<td>Stepper Motor Driver 5pcs</td>
<td>Amazon</td>
<td>1</td>
<td>21.49</td>
</tr>
<tr>
<td>Stepper Motor Wires</td>
<td>Amazon</td>
<td>1</td>
<td>8.99</td>
</tr>
<tr>
<td>12V Power Supply</td>
<td>Amazon</td>
<td>2</td>
<td>29.80</td>
</tr>
<tr>
<td>Pressure Sensors 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Analog to Digital Converter</td>
<td>Amazon</td>
<td>3</td>
<td>13.65</td>
</tr>
<tr>
<td>Ultrasonic Sensor</td>
<td>Amazon</td>
<td>1</td>
<td>13.99</td>
</tr>
<tr>
<td>RoboDK License</td>
<td>RoboDK</td>
<td>1</td>
<td>145.00</td>
</tr>
<tr>
<td>Electrical Hardware Hit</td>
<td>Amazon</td>
<td>1</td>
<td>18.49</td>
</tr>
<tr>
<td>Jumper Wires</td>
<td>Amazon</td>
<td>1</td>
<td>25.95</td>
</tr>
<tr>
<td>Metal Gripper</td>
<td>Xometry</td>
<td>1</td>
<td>1361.29</td>
</tr>
<tr>
<td>Plastic Gripper</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>88.11</td>
</tr>
<tr>
<td>Handle</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>9.97</td>
</tr>
<tr>
<td>Gripping Material</td>
<td>Amazon</td>
<td>2</td>
<td>25.98</td>
</tr>
<tr>
<td>Nema Stepper Motors</td>
<td>Amazon</td>
<td>5</td>
<td>84.15</td>
</tr>
<tr>
<td>M3 Screws</td>
<td>Amazon</td>
<td>1</td>
<td>11.49</td>
</tr>
<tr>
<td>Monitor</td>
<td>Best Buy</td>
<td>1</td>
<td>299.99</td>
</tr>
<tr>
<td>Conveyor Belt</td>
<td>Vevor</td>
<td>1</td>
<td>248.99</td>
</tr>
<tr>
<td>Assorted Zip Ties Pack</td>
<td>Amazon</td>
<td>1</td>
<td>15.95</td>
</tr>
<tr>
<td>Poster</td>
<td>Print Shop</td>
<td>1</td>
<td>110.00</td>
</tr>
<tr>
<td>Shirts</td>
<td>Engineering College</td>
<td>5</td>
<td>150.00</td>
</tr>
<tr>
<td>Raspberry Pi HQ Camera</td>
<td>Vilros</td>
<td>1</td>
<td>50.00</td>
</tr>
<tr>
<td>6mm Camera Lens</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Total (Excluding Tax and Shipping)</td>
<td></td>
<td></td>
<td>3114.43</td>
</tr>
<tr>
<td>Total (Including Tax and Shipping)</td>
<td></td>
<td></td>
<td>3358.96</td>
</tr>
<tr>
<td>Category</td>
<td>Cost (USD)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electrical/Software</td>
<td>610.73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mechanical</td>
<td>1579.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>808.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optical</td>
<td>75.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3073.73</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical/Software</td>
<td>610.73</td>
</tr>
<tr>
<td>Mechanical</td>
<td>1579.02</td>
</tr>
<tr>
<td>Optical</td>
<td>75.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2264.75</strong></td>
</tr>
</tbody>
</table>

**Total (Tax, Shipping)** $3270.2

**Total (Tax, Shipping)** $2461.22
**Pick Rate Demonstration:**

**Requirement and Justification:**

2.3.3 Pick rate: The ARG shall have a pick rate of at least 30 picks per minute. (0.6 seconds/grip)

With the high volume of products moving down the line at Unilever manufacturing plants, time is money. The more objects that can be gripped in a given time period, the more products make it out the door. This being the case it was important that the gripper could facilitate 30 picks per minute. A ‘pick’ is defined as the process of grabbing the object, moving it to its new location, setting it down, and returning to the initial position to grab the next object. In this process the majority of the time, 1.4 seconds in the case of Unilever robot arms, is spent moving back and forth. This leaves 0.6 seconds for the gripper to close around the object and open again to release it. Since we were without an assembly line to test this full scale in its final environment, we timed the opening and closing process for each of the objects and recorded these values. So long as it was able to open and close securely around an object in less than 0.6 seconds, it was considered a successful demonstration. The demonstration sheet with these results is displayed below.

<table>
<thead>
<tr>
<th>2.2.3 Acceptance Test Data Sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced ATP Paragraph Number: N/A</td>
</tr>
<tr>
<td>Analysis Referenced: N/A</td>
</tr>
<tr>
<td>Name of Test: Pick Rate Test</td>
</tr>
<tr>
<td>Unit Under Test (UUT):</td>
</tr>
<tr>
<td>Name: Mechanical Gripper</td>
</tr>
<tr>
<td>Part Number: 104100,104302</td>
</tr>
<tr>
<td>Results (Pass/Fail): Pass</td>
</tr>
<tr>
<td>Recording of Demonstration Measurement:</td>
</tr>
<tr>
<td>Object 1: SOAP</td>
</tr>
<tr>
<td>Object 2: DEODORANT</td>
</tr>
<tr>
<td>Object 3: BODYWASH</td>
</tr>
<tr>
<td>Object 4: SPRAY</td>
</tr>
<tr>
<td>Object 5: TEA</td>
</tr>
<tr>
<td>Requirement (with tolerances):</td>
</tr>
<tr>
<td>Requirement (with tolerances): Must be less than 0.6 seconds</td>
</tr>
<tr>
<td>Time (seconds):</td>
</tr>
<tr>
<td>Object 1: 0.39</td>
</tr>
<tr>
<td>Object 2: 0.42</td>
</tr>
<tr>
<td>Object 3: 0.55</td>
</tr>
<tr>
<td>Object 4: 0.51</td>
</tr>
<tr>
<td>Object 5: 0.41</td>
</tr>
</tbody>
</table>
Adaptability Demonstration:

Requirement and Justification:

2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products

The gripper must be able to grip five unique Unilever products that vary in shape and weight: Axe deodorant, Dove bodywash, Lipton tea, Dove bar soap, and Axe body spray. While the overall performance of the system is tested in the performance test, this preceding demonstration was simply meant to show that the gripper was capable of picking up each object. The demonstration sheet and images of each object being held by the gripper are displayed below.

<table>
<thead>
<tr>
<th>2.4.1 Acceptance Test Data Sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced ATP Paragraph Number: N/A</td>
</tr>
<tr>
<td>Analysis Referenced: N/A</td>
</tr>
<tr>
<td>Name of Test: Adaptability Demonstration</td>
</tr>
<tr>
<td>Unit Under Test (UUT):</td>
</tr>
<tr>
<td>Name: Adaptability Demonstration</td>
</tr>
<tr>
<td>Part Number: 104100, 104300</td>
</tr>
<tr>
<td>Results (Pass/Fail): Pass</td>
</tr>
<tr>
<td>Recording of Demonstration Measurement:</td>
</tr>
<tr>
<td>Could gripper pick up object?:</td>
</tr>
<tr>
<td>Object 1: BODYWASH</td>
</tr>
<tr>
<td>Object 2: DEODORANT</td>
</tr>
<tr>
<td>Object 3: SOAP</td>
</tr>
<tr>
<td>Object 4: SPRAY</td>
</tr>
<tr>
<td>Object 5: TEA</td>
</tr>
</tbody>
</table>

Computations and Notes:

Gripper is capable of grabbing all objects.
Dove Bodywash

Dove Bar Soap

Axe Body Spray

Lipton Tea

Axe Deodorant
Handle Demonstration:

Requirement and Justification:

2.4.2 Handle: the ARG will have a handle for demonstration

Since the team does not have a robot arm a handle is needed to facilitate demonstration of the system. Using this handle, we will be able manually position the gripper as accurately as we can. The SolidWorks file showing the handle is displayed below.
Safety Demonstration:

Requirement and Justification:

2.4.3 Safety: The ARG shall have a shut-down mechanism for emergency situations

In order to make the system as safe as possible, a shut-off button has been implemented. Upon the press of the button, the system will stop all processes immediately. The wire diagram for this button is shown below.

While very simple, this setup is an effective input method initiate the shut-off.
TDP summary

1 Scope

The document shall include an overview of the system that the software is part of, identification of the software configuration items, and a document overview.

The adaptive robot gripper will use software mainly in three different areas: 1) movement control, 2) force feedback, and 3) object recognition/location. To allow a mechanical gripper the ability to adapt and move any product, it must have the ability to recognize the product that it is picking up and understand the environment in which it operates. The system can gather this information which will allow it to adjust the gripper to specific products and move them from one desired location to a final desired location. The main inputs from which our software will gather information are the 1) camera, which is used for object recognition, 2) force resistors, which is used for force feedback, and 3) ultrasonic distance sensor, which is used for object location. The system uses the camera to visually identify various products through object recognition so it understands the product that it will interact with. The trained image algorithm will figure out how the specific product needs to be picked up from preprogramed information gathered for all the different products that the gripper will interact with. The force resistors will provide constant feedback as the gripper interacts with the product, and the control will decide whether to apply force as needed. The ultrasonic distance sensor can identify the location of the object relative to the edge of the conveyer belt, and this information will be passed to the software to process and move the arm to the correct position. The main outputs are 1) stepper motor, which will open and close the gripper to the correct size depending on the size/weight/density of the object, and 2) robot arm, which will move the gripper to the correct location and orientation. In our case, we used emulation software to emulate the robot arm, as we were not able to get a robot arm.

The purpose of this document is to show the intention of the project software, explain how the created system works, and to provide evidence and code that shows how it works. This document will go over the reference documents, CSCI-Wide design decisions, CSCI architectural design, CSCI detailed design, requirement traceability, and additional notes from the Adaptive Robot Gripper team. With the information from this document, any user should be able to understand how the software works without having to look at the actual code or watching the device function.

Changes in Version3 of the TDP include a copy of the final integrated software, Raspberry Pi setup instructions, and how to train a YOLOv5 image recognition neural network model.
Acceptance Test Results

Object Identification Test:

Requirement and Justification:

2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolov5 neural network model (95% accuracy)

To reach the 95% threshold outlined above, the ability of our camera system to correctly identify each of our test objects must be tested.

Procedure:

1. Set up a controlled imaging environment. Mount the camera in a stationary position such that the viewing background is uniform and with appropriate lighting.
2. Place an object in the camera’s field of view.
3. Read the software output to confirm the object was correctly identified with at least 70% confidence.
4. Complete steps 3 and 4 at least 3 times for each of the 5 objects. Record the results.

The justification for the 70% passing metric comes from the software training process we used to identify our objects. We used over 1000 images to train the software and any object that was identified with a confidence of 70% or more was always a correct identification.
### 2.1.1 Acceptance Test Data Sheet

Referenced ATP Paragraph Number: 2.1.1

Analysis Referenced: N/A

Name of Test: Object ID

Unit Under Test (UUT):
Name: Visualization System
Part Number: 101000, 102101

Results (Pass/Fail): **Pass**  
Date of Test: **2/15/22**

**Recording of Test Measurement:**
**Object 1:** BODYWASH
**Object 2:** DEODORANT
**Object 3:** SOAP
**Object 4:** SPRAY
**Object 5:** TEA

**Requirement (with tolerances):**
Must be recognized with >70% confidence

**Readout 1:** 75, 86, 82 – Pass  
**Readout 2:** 79, 85, 87 – Pass  
**Readout 3:** 76, 83, 86 – Pass  
**Readout 4:** 91, 88, 92 – Pass  
**Readout 5:** 89, 87, 90 – Pass

Smallest Margin: 5%  
Average Margin: 15%  
PASS

Computations and Notes:

* 70% metric was decided through testing. For our 1000+ training image any that had 70%+ confidence correctly identified the object.
Distance Detection Test:

Requirement and Justification:

2.2.2 Distance Detection: The ultrasonic sensor shall report object distances. (+/- 5mm)

In order to give location data to the robot arm responsible for positioning the gripper, a distance sensor will determine objects position on the conveyor belt. Since the grip heights have been preprogrammed and the conveyor will stop in the same position each time, this sensor only needs to operate on one axis, the lateral position of the object on the belt. The following test was conducted to ensure the accuracy of this detector.

Procedure:

1. Set up the ultrasonic sensor in a stationary position with at least 1.5 meters of space in front of it.
2. Place an object some distance in front of the ultrasonic sensor.
3. Measure the distance from the sensor to the object manually. (Metric measurement is preferred to avoid unit conversions)
4. Read the software output with sensed measurement.
5. Compare outputs from steps 4 and 5. If they are within 5mm of one another, this trial is a pass.
6. Complete steps 2-5 at least 5 times. Record the results.
## 2.2.2 Acceptance Test Data Sheet

Referenced ATP Paragraph Number: 2.2.2

Analysis Referenced: N/A

Name of Test: Distance Test

Unit Under Test (UUT):
Name: Ultrasonic Sensor
Part Number: 105200

<table>
<thead>
<tr>
<th>Results (Pass/Fail): Pass</th>
<th>Date of Test: 2/17/22</th>
</tr>
</thead>
</table>

**Recording of Test Measurement:**

<table>
<thead>
<tr>
<th>Measured Distance (mm)</th>
<th>Requirement (with tolerances):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 304.8</td>
<td>Must be with +/- 5 mm</td>
</tr>
<tr>
<td>2. 254</td>
<td>Sensed Distance (mm)</td>
</tr>
<tr>
<td>3. 203.2</td>
<td>1. 302.8 (-2 mm): Pass</td>
</tr>
<tr>
<td>4. 101.6</td>
<td>2. 254.4 (-2.6 mm): Pass</td>
</tr>
<tr>
<td>5. 50.8</td>
<td>3. 202.8 (-.5 mm): Pass</td>
</tr>
</tbody>
</table>

Smallest Margin: 2.4mm
Average Margin: 2.96mm

**Computations and Notes:**

1 Inch = 2.54 cm, 1 cm = 10 mm

PASS
Force Sensor Calibration Test:

Requirement and Justification:

2.2.3 Force Sensor: The ARG shall be able to apply the appropriate force to the object by having the ability to accurately measure the force it is currently applying. (+/-1) N

The initial purpose for the FSR was to ensure the object was not gripped too tightly or too loosely to avoid crushing or dropping the object. From the test sheet below it can be seen that this part failed to meet the intended specification. This being the case, we repurposed the FSR to only monitor whether or not the object has been dropped so the system could be shut down in case of failure. Despite the FSR no longer setting the upper limit of grip force, crushing the objects remains a non-issue since the motors used in the adaptive robot gripper are not strong enough to damage any of the objects under test. Even with this new purpose, the force sensitive resistor used for the gripper is inadequate. For future iterations of the gripper, a new one should be selected.

Procedure:

1. Set up the force sensitive resistor (FSR) stationarily on a flat surface.
2. Use a scale to measure the mass of your test object. (54 grams for the data below)
3. Convert measured mass to gravitational force using equation 1 on the test sheet below.
4. Place test object on the sensor.
5. Read the software output with sensed measurement in bits.
6. Convert bit value to force value using equation 2 on the test sheet below.
7. Compare outputs from steps 3 and 6. If they are within 1N of one another, this trial is a pass.
8. Complete steps 2-5 at least 5 times. Record the results.
## 2.2.3 Acceptance Test Data Sheet

| Referenced ATP Paragraph Number: | 2.2.2 |
| Analysis Referenced: | N/A |
| Name of Test: | Force Calibration Test |
| Unit Under Test (UUT): | Name: Force Sensor | Part Number: 105100 |
| Results (Pass/Fail): | Fail | Date of Test: 4/12/22 |

### Recording of Test Measurement:

<table>
<thead>
<tr>
<th>Measured force (N)</th>
<th>Requirement (with tolerances):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each weight is 54g</td>
<td>Must be with 1 N</td>
</tr>
<tr>
<td>1. 216g -&gt; 2.12N; 22bits</td>
<td>Sensed Force (N)</td>
</tr>
<tr>
<td>2. 270g -&gt; 2.65N; 28bits</td>
<td>1. 45 bits -&gt; 4.31N (+2.19N): Fail</td>
</tr>
<tr>
<td>3. 324g -&gt; 3.18N; 33bits</td>
<td>2. 51 bits -&gt; 4.89N (+2.23N): Fail</td>
</tr>
<tr>
<td>4. 378g -&gt; 3.71N; 38 bits</td>
<td>3. 22 bits -&gt; 2.11N (-1.07N): Fail</td>
</tr>
<tr>
<td>5. 432g -&gt; 4.24N; 44 bits</td>
<td>4. 27 bits -&gt; 2.59N (-1.12N): Fail</td>
</tr>
<tr>
<td>5. 432g -&gt; 4.24N; 44 bits</td>
<td>5. 24 bits -&gt; 2.30N (-1.94N): Fail</td>
</tr>
</tbody>
</table>

Average Error: 1.71N
Average % Error: 71%

FAIL

### Computations and Notes:

1. \( N = m \times 9.81 \text{ms}^{-1} \)
2. \( N = (10 \text{kg} \times \text{nbits} \times 9.81 \text{ms}^{-1}) / 1024 \)
Engineering Change Request

Date: 19 April 2022

Team Number 22053  Project Name: Adaptive Robot Gripper

Title of Change: Force Sensor Requirement 2.2.3

Documents Affected (check all that apply):

<table>
<thead>
<tr>
<th>Doc Type</th>
<th>Document Number</th>
<th>Document Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ SOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ SRD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ SDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ TP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ EVMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Drawing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ Other</td>
<td></td>
<td>FAR and Final Report</td>
</tr>
</tbody>
</table>

Description of Change: Requirement 2.2.3 states, “Force Sensor: The ARG shall be able to apply the appropriate force to the object by having the ability to measure the force it is currently applying.” While we will still use the force sensor, its purpose will be to check whether or not the product has been grabbed. This will remove the force sensor feedback loop for the gripper and the information of whether the gripper has picked the item will only be sent to the GUI. Due to this change, we would also like to change this requirement from being a test to a display.

Need for Change: The force sensors that we have been working with have not been able to read the force being applied to the products consistently and accurately.

Impact: This will remove the force sensor feedback from our system and replace it with a check on whether or not the gripper has picked up the item. Since this is intended to be on an assembly line and items are on a conveyor belt, the gripper and robot arm will not be affected by this signal that tells the software if the item has been grabbed.

Substantiation: This is just to remove the force sensor feedback loop. Our system is still able to perform with out it and not damage the items. The only change is that the gripper will lose some precision with the removal of this sensor feedback. The system will rely upon the pre-coded data to properly grab the products that it recognizes.
**Force Sensor Drop Demonstration:**

**Requirement and Justification:**

2.2.3 Force Sensor: The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips

The initial purpose for the FSR was to ensure the object was not gripped too tightly or too loosely to avoid crushing or dropping the object. From the test sheet above it can be seen that this part failed to meet the intended specification. This being the case, we repurposed the FSR to only monitor whether or not the object has been dropped so the system could be shut down in case of failure. Despite the FSR no longer setting the upper limit of grip force, crushing the objects remains a non-issue since the motors used in the adaptive robot gripper are not strong enough to damage any of the objects under test. Even with this new purpose, the force sensitive resistor used for the gripper is inadequate due to inconsistent readings. For future iterations of the gripper, a new one should be selected.

**Procedure:**

For this demonstration, the gripper holds an object which is then manually removed from the fingers while it is still meant to be held. The force values output by the sensor are monitored during this process. A consistent change from a high to low threshold must be observed so these thresholds can be coded into the project as ‘held’ and ‘dropped’ states.
## 2.2.3 Acceptance Test Data Sheet

Referenced ATP Paragraph Number: 2.2.2

Analysis Referenced: N/A

Name of Test: Force Sensor Drop Demonstration

<table>
<thead>
<tr>
<th>Unit Under Test (UUT):</th>
<th>Name: Force Sensor</th>
<th>Part Number: 105100</th>
</tr>
</thead>
</table>

Results (Pass/Fail): **Fail**  
Date of Test: **4/12/22**

### Recording of Test Measurement:

<table>
<thead>
<tr>
<th>Measured force while held (bits)</th>
<th>Requirement (with tolerances):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 122</td>
<td><strong>Must show demonstrable drop in force output</strong></td>
</tr>
<tr>
<td>2. 100</td>
<td></td>
</tr>
<tr>
<td>3. 131</td>
<td></td>
</tr>
<tr>
<td>4. 85</td>
<td></td>
</tr>
<tr>
<td>5. 90</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measured force while dropped(bits)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 65</td>
<td>FAIL</td>
</tr>
<tr>
<td>2. 78</td>
<td></td>
</tr>
<tr>
<td>3. 96</td>
<td></td>
</tr>
<tr>
<td>4. 105</td>
<td></td>
</tr>
<tr>
<td>5. 82</td>
<td></td>
</tr>
</tbody>
</table>

### Computations and Notes:

Despite a general downtrend in detected force, there is too much overlap between the outputs between held and dropped for a reliable distinction to be made.
Performance Test:

Requirement and Justification:

2.3.2 Performance: The ARG shall have a success rate of 95%.

For a product that will be used in a production line handling up to 30 objects per minute, it is important to have a success rate that is as close to perfect as possible. Even with a 99.9% success rate the gripper would fail more than 14 times in an 8-hour period. This being the case, we were initially tasked with getting as close to 100% as possible, but with less than a year to develop the gripper and the fact that 100% accuracy is not something we could measure in a way that is statistically significant, Unilever agreed to reduce it to 95% so that we could properly measure it. To test this metric, we have to test the system in its entirety. Therefore, a success can only be recorded if all of the following steps are successful:

1. The distance detector must trigger when an object is present to start the process.
2. The camera must image the object for the software to identify.
3. The object must be identified correctly by the software.
4. The object must be moved down the conveyor to the gripping plane.
5. The gripper must pick up the object successfully.
6. The gripper must release the object in the placement zone.

Procedure:

1. Start the code.
2. Place an object in the detection plane.
3. Wait for camera to identify the object.
4. Activate the conveyor belt.
5. Position the gripper in the correct orientation for the object.
6. Move object to placement zone.
7. Repeat above steps 5 times for each object.

The results for this test are displayed on the datasheet below.
## 2.3.2 Acceptance Test Data Sheet

Referenced ATP Paragraph Number: N/A

Analysis Referenced: N/A

Name of Test: Adaptability Demonstration

Unit Under Test (UUT):
Name: Performance Test
Part Number: Whole System

Results (Pass/Fail): **Pass**

<table>
<thead>
<tr>
<th>Recording of Demonstration Measurement:</th>
<th>Results:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1: <strong>BODYWASH</strong></td>
<td>System must pass 95% of the time</td>
</tr>
<tr>
<td>Object 2: <strong>DEODORANT</strong></td>
<td>Object 1: Yes 5/5</td>
</tr>
<tr>
<td>Object 3: <strong>SOAP</strong></td>
<td>Object 2: Yes 5/5</td>
</tr>
<tr>
<td>Object 4: <strong>SPRAY</strong></td>
<td>Object 3: Yes 5/5</td>
</tr>
<tr>
<td>Object 5: <strong>TEA</strong></td>
<td>Object 4: Yes 4/5</td>
</tr>
<tr>
<td></td>
<td>Object 5: Yes 5/5</td>
</tr>
<tr>
<td></td>
<td>Pass rate: 96%</td>
</tr>
</tbody>
</table>

**Computations and Notes:**
- System pass is defined on the previous page.
- Drops from user error were omitted.
## Final Budget

**Table 1:** Project Components Cost Separated by Category (Excluding Display Costs)

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical/Software</td>
<td>633.51</td>
</tr>
<tr>
<td>Mechanical</td>
<td>1580.99</td>
</tr>
<tr>
<td>Optical</td>
<td>75.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2289.50</strong></td>
</tr>
<tr>
<td><strong>Total (Tax, Shipping)</strong></td>
<td><strong>2507.93</strong></td>
</tr>
</tbody>
</table>

**Table 2:** Project Cost Separated by Category (Including Display)

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical/Software</td>
<td>633.51</td>
</tr>
<tr>
<td>Mechanical</td>
<td>1580.99</td>
</tr>
<tr>
<td>Display</td>
<td>824.93</td>
</tr>
<tr>
<td>Optical</td>
<td>75.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3114.43</strong></td>
</tr>
<tr>
<td><strong>Total (Tax, Shipping)</strong></td>
<td><strong>3358.96</strong></td>
</tr>
</tbody>
</table>
### Table 3: Project Components Cost

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 8GB RAM</td>
<td>Labists</td>
<td>2</td>
<td>302.98</td>
</tr>
<tr>
<td>Camera Flex Cable</td>
<td>Amazon</td>
<td>1</td>
<td>8.79</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Amazon</td>
<td>1</td>
<td>12.99</td>
</tr>
<tr>
<td>Buttons 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>6.39</td>
</tr>
<tr>
<td>Stepper Motor Driver 5pcs</td>
<td>Amazon</td>
<td>1</td>
<td>21.49</td>
</tr>
<tr>
<td>Stepper Motor Wires</td>
<td>Amazon</td>
<td>1</td>
<td>8.99</td>
</tr>
<tr>
<td>12V Power Supply</td>
<td>Amazon</td>
<td>2</td>
<td>29.80</td>
</tr>
<tr>
<td>Pressure Sensors 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Analog to Digital Converter</td>
<td>Amazon</td>
<td>3</td>
<td>13.65</td>
</tr>
<tr>
<td>Ultrasonic Sensor</td>
<td>Amazon</td>
<td>1</td>
<td>13.99</td>
</tr>
<tr>
<td>RoboDK License</td>
<td>RoboDK</td>
<td>1</td>
<td>145.00</td>
</tr>
<tr>
<td>Electrical Hardware Hit</td>
<td>Amazon</td>
<td>1</td>
<td>18.49</td>
</tr>
<tr>
<td>Jumper Wires</td>
<td>Amazon</td>
<td>1</td>
<td>25.95</td>
</tr>
<tr>
<td>Metal Gripper</td>
<td>Xometry</td>
<td>1</td>
<td>1361.29</td>
</tr>
<tr>
<td>Plastic Gripper</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>88.11</td>
</tr>
<tr>
<td>Handle</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>9.97</td>
</tr>
<tr>
<td>Gripping Material</td>
<td>Amazon</td>
<td>2</td>
<td>25.98</td>
</tr>
<tr>
<td>Nema Stepper Motors</td>
<td>Amazon</td>
<td>5</td>
<td>84.15</td>
</tr>
<tr>
<td>M3 Screws</td>
<td>Amazon</td>
<td>1</td>
<td>11.49</td>
</tr>
<tr>
<td>Monitor</td>
<td>Best Buy</td>
<td>1</td>
<td>299.99</td>
</tr>
<tr>
<td>Conveyor Belt</td>
<td>Vevor</td>
<td>1</td>
<td>248.99</td>
</tr>
<tr>
<td>Assorted Zip Ties Pack</td>
<td>Amazon</td>
<td>1</td>
<td>15.95</td>
</tr>
<tr>
<td>Poster</td>
<td>Print Shop</td>
<td>1</td>
<td>110.00</td>
</tr>
<tr>
<td>Shirts</td>
<td>Engineering College</td>
<td>5</td>
<td>150.00</td>
</tr>
<tr>
<td>Raspberry Pi HQ Camera</td>
<td>Vilros</td>
<td>1</td>
<td>50.00</td>
</tr>
<tr>
<td>6mm Camera Lens</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Total (Excluding Tax and Shipping)</td>
<td></td>
<td></td>
<td>3114.43</td>
</tr>
<tr>
<td>Total (Including Tax and Shipping)</td>
<td></td>
<td></td>
<td>3358.96</td>
</tr>
</tbody>
</table>
### Table 3: Metal Gripper Parts Cost Breakdown

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty.</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grip Plate</td>
<td></td>
<td>4</td>
<td>127.72</td>
</tr>
<tr>
<td>Link</td>
<td></td>
<td>15</td>
<td>288.75</td>
</tr>
<tr>
<td>Finger</td>
<td></td>
<td>4</td>
<td>298.40</td>
</tr>
<tr>
<td>Gear 1</td>
<td></td>
<td>4</td>
<td>201.80</td>
</tr>
<tr>
<td>Palm</td>
<td></td>
<td>2</td>
<td>183.64</td>
</tr>
<tr>
<td>Drive Gear</td>
<td></td>
<td>2</td>
<td>260.98</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>1361.29</td>
</tr>
<tr>
<td>Tax</td>
<td></td>
<td></td>
<td>111.62</td>
</tr>
<tr>
<td>Shipping</td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>Total (Tax and Shipping)</td>
<td></td>
<td></td>
<td>1472.91</td>
</tr>
</tbody>
</table>

### Table 4: Plastic Gripper Parts Cost Breakdown

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty.</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SARAH 1</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>5.30</td>
</tr>
<tr>
<td>SARAH 2</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>11.80</td>
</tr>
<tr>
<td>Modified Parts 1</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>5.00</td>
</tr>
<tr>
<td>Modified Parts 2</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>1.80</td>
</tr>
<tr>
<td>SARAH 3</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>23.50</td>
</tr>
<tr>
<td>SARAH 4</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>39.70</td>
</tr>
<tr>
<td>Modified Parts 3</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>1.01</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>88.11</td>
</tr>
<tr>
<td>Tax</td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>Shipping</td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>Total (Tax and Shipping)</td>
<td></td>
<td></td>
<td>88.11</td>
</tr>
</tbody>
</table>
Table 5: Stepper Motor Cost Breakdown

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty.</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>46Ncm Torque</td>
<td>Amazon</td>
<td>2</td>
<td>35.18</td>
</tr>
<tr>
<td>59Ncm Torque</td>
<td>Amazon</td>
<td>2</td>
<td>27.98</td>
</tr>
<tr>
<td>65Ncm Torque</td>
<td>Amazon</td>
<td>1</td>
<td>20.99</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>84.15</td>
</tr>
</tbody>
</table>
Lessons Learned
Lucas Hawley – Communication

Communication is vital in engineering, in 1907 the Quebec Bridge collapsed during construction killing over eighty workers due to a lack of communication of an error. Obviously, the stakes at play in a Senior Engineering Project are much lower and there was little risk of injuries to us, but communication still held a large role in our project. Things changed greatly between the sponsor team that started the project and the one that remained for much of the project. It was vital to us to keep communication flowing as ideas for the plan of the project and scope changed over time.

The first lesson to that effect that was learn was how important regular meeting are specially if the team you are working with aren’t in the same area as you. There were large concerns for this project that stemmed from irregular emails asking different questions being randomly sent as needed. This quickly proved ineffective as the projects stakeholders weren’t on the same page. Once regular meeting was established time change and distance were still factors but the project began to move much smoother.

The second lesson was the one on team management. People are human, situations are situational, and murphy’s law comes for us all. When I say people are human, I mean to say things come up. Communicating with the team to know when exams or reports are due allowed us to shift workload flexibly as at the end of day the team is made up with students with many variables outside of this project. Situations for these things couldn’t be just plugged in and expect the same outcome as again many outside variables could play their hand on the project. Finally, murphy’s law needs to be accounted for and communicated which was demonstrated when the hour and a half time slot for preparing the project on Design Day became a three-hour grind to find anyway to get the demonstration to work. Through communicating and sticking to our roles we were able to work the issues and deliver a wonderful demonstration and presentation on the day it mattered most.

Communication is more than a goal but a necessity for a project like this and the lessons learned here will inform all of our careers as we move forward into industry.
Connor Nagore - Redundancy

The lesson that I learned from the Adaptive Robot Gripper Senior Design project was the importance of redundancy. In terms of an engineering project, this means that we were making sure that the team had extra parts and contingency plans for each part of the project. This is significant because it allows the team to be better prepared for an accident, extra parts reduce the chance that a broken or malfunctioning part does not ruin the schedule or budget when the project has progressed further, and contingency plans reduce the chance that problems cripple the project schedule.

There were many unforeseen problems that appeared throughout this project such as accidents. While doing our best to work according to the safety of the team members and equipment, there was still a possibility that an accident may occur. For example, one of the team members accidentally left the main gripper in their car causing deformation. Because we had a second gripper printed, we were able to continue with our testing without much delay. Having extra parts or contingency plans that allow the project to recover from most accidents allows the schedule and budget to be minimally affected by such unforeseen events.

Ordering extra parts reduces the chance that malfunctioning parts affect the budget and schedule of a project. Not all parts are going to work according to their data sheet and returning and reordering takes a lot of time and possibly some money. A good example of this would be an event that occurred on Design Day. Our Raspberry Pi would not connect to any of the monitors, which would have crippled our presentation. We did have a second Raspberry Pi that we were able to switch and get our system working again before the judges came to our table. Ordering extra parts may slightly increase the cost at the time of order, but it allows the product to avoid situations that the project may have to pay in both time and money if a part were to malfunction.

Finally, when working with the logistics of the project, having a few contingency plans reduces the chance that any problem cripples the schedule or budget. Some plans do not work out, like parts not coming in on time or designs not working like they would in theory. If a team does not have multiple plans before they approach problem solving, then an event that changes the direction of the project could cripple the schedule and budget because the team would have to go back to redesigning a part of the project or the whole project itself. Having contingency plans allows a team to be fluid and adjust their approach as problems arise.

The importance of this lesson of redundancy was learned through a few various events throughout this project. There were times that we had a part fail on us after we pushed then to their limits or through a few accidents where parts were ruined by being dropped and we had some parts melt. Designs we made did not work as we had hoped, and we had to change our approach to different parts of our project. Allowing for contingency plans and ordering extra parts allowed us to immediately act and adjust our direction in order to continue to make progress to the project and fix the problem that came up.
Max - Time Management

With a large project that lasts nearly a year, keeping a schedule and following it is extremely important. Unlike many of the other classes I’ve taken at the University of Arizona where the professor sets the calendar, we were required to make our own schedule between each of our major deadlines.

In addition to teaching us a useful skill, the creation of our schedule was also helpful for keeping our team accountable. Utilizing SmartSheets was great because it clearly set out each teammate’s assignments at any given time, so we never devoted too many resources to one thing or forgot a deadline.

SmartSheets were also helpful for ensuring we had enough time to complete all of our work. Setting personal deadlines prior to official ones allowed us to overcome any unforeseen obstacles or failed tests. One example of this was our CDE 1. The camera setup we used initially was meant to identify and locate the objects from an overhead view. This angle was not sufficient to distinguish the objects and the location values it gave were inaccurate. This potentially could have been a major setback had we lacked the time to fix it. Fortunately, we had the time to overhaul the system by adding a distance sensor for location and changing the camera angle.

Proper time management was key for this project and without it we certainly would not have completed our project in time. This is a lesson that will be useful in the future as well and was invaluable throughout senior design.
Joe – Software Documentation

The complexity of this project, with so many moving parts, integrating both mechanical and software components into a single product can be very difficult without proper documentation. Having reference documents ready at hand allows for quick setup, testing, and debugging of components without the headache of trying to remember what website you last used or where you left off on the development of code.

One important component of the software documentation was having updated wiring diagrams. Our Raspberry Pi needed to have fans to cool the CPU, distance sensor wiring, camera FPC wiring, force sensor wiring, stepper motor wiring, and stop button wiring. Having an updated PIN diagram greatly decreased set up time and made sure there were no crossing wires during the full integration.

Another important aspect of documentation specifically for software is using GitHub to handle version control and to allow other members of the team to easily access the most updated code. Being the software lead of the team meant that I was primarily in charge of all the code, and only storing the program locally not only made it difficult for other members to test out specific components, but it also posed a great danger if the local machine were to fail and all files in development were lost.

It is very important to have documentation of the tolerances, data sheets, and purchase links for all our components. One time the analog to digital converter melted the breadboard because we plugged it into the 5V power supply instead of the 3.3V power supply. From the data sheets, we also learned that the stepper motor can go all the way down to 1/32 of a step, allowing for fine motor control. We used this to turn the motor accurately, which helped with properly gripping the objects. Having the data sheets lets us know what each component is capable of and having purchase links allows us to quickly buy replacement parts when needed.

Lastly, software documentation allows for a quick and easy transition from one team to another. If there is a new team that wanted to expand on this project, they can easily set everything up from following the documentation. One very important thing to have for the future team is how to install the OpenCV library on the Raspberry Pi so that the code will run properly. It is a very convoluted process, and without any documentation, it would take a beginner a week to figure out.
Riyadh – Testing

The most important lesson that I learned from the adaptive robot gripper is the importance of testing. We always need to start testing and building physical prototypes at an early stage of the project. Some things in the build of the adaptive robot gripper need a physical prototype and to be tested to pin what is wrong with them.

No matter how much we design and work on the AutoCAD we can’t know the exact tolerances of the machine that will print our prototype and we always miss a few mm here and there. The biggest problem we faced was that we printed all our prototypes using the university printing services and those services have a lot of 3d printing machines and for each order we get a different machine with different tolerances, so we had to depend on trial and error until we got everything working with the physical prototype of the gripper.

Another important point is the integration of the software whether it was motor, camera or sensor software and sometimes when we integrate all of this together, we face problems we never thought we will face. Testing early and many times will make us fail fast and recover faster, testing makes us be more familiar with our systems and how to integrate them perfectly. Testing early and a lot doesn’t just help us with the integration and making our prototype work perfectly it also makes us know how to optimize our performance to preform our task even more accurately and efficiently.
Next Steps

Before discussing the next steps, it is important to talk about possible improvements that can be made to the current version of this project. These are also discussed because these items will be discussed in the next steps section. The improvements listed below are in order from most important to accomplish first to least important. Their location in the next steps when continuing this project will be discussed in the Next Steps section.

Tolerancing

The main problem with the gripper that was produced by the Adaptive Robot Gripper team had problems with tolerancing. The 3-D printed plastic grippers had tolerances that allowed the grip to occasionally give some slack after the motor tightened around a product. Due to the expense of ordering metal pins from Xometry, 3-D plastic pins were used for the assembly of the aluminum version of the gripper. The tolerances that resulted from this caused the gear links to be able to shift closer together and the gears would lock together.

Printed Circuit Board (PCB)

When integrating the systems of the Adaptive Robot Gripper, all of the electrical components were plugged together with a bread board. This setup is still a prototype and was the most progress that could be made, but groups continuing this project could take the wiring diagrams and make a PCB board to make the set up more organized and professional.

More Accurate Force Sensors

During the implementation of the designs of the Adaptive Robot Gripper, the design for force sensor feedback had to be removed because of the resistance sensors that were tested. The resistors used did not provide accurate or consistent feedback for forces that we attempted to measure with them. To reimplement a force sensor feedback, teams continuing the project will need to experiment with different forms of force sensors from the resistance sensor used in the testing for this project. A more precise and accurate sensor would allow the design for force sensor feedback to be reincorporated in this design.

Gripper Fingertips

Since Unilever has a diverse number of products that belong to their companies that they control. Just a few examples of these companies are Ben and Jerry’s, Dove, Axe, and Lipton. Of these companies, 5 different products were used: Axe body spray, Axe deodorant, Dove bodywash, Dove bar soap, and Lipton Tea.

In experimenting with these products, the Adaptive Robot Gripper team found that the products had varying geometries, some being simple and others being complex. To adjust for these, the physical
The gripper was designed with a soft, high friction material along with elongated fingertips to increase the surface area of the fingertips that touches the products. While our gripper design was successful in picking up all the products that the Adaptive Robot Gripper team chose to pick up, there is limited ways that the items could be picked up because the fingertips were ridged.

To improve this, the Adaptive Robot Gripper team suggest improvements that could increase the flexibility of the fingertips so that the gripper can pick up more items with complex geometries. The fingertips could be connected to the finger by an axle with a spring in order to return it to a normal position after gripping a product whose sides are more angled rather than being completely vertical. The axle joint will allow the fingertip to change its pitch to allow more surface area of the high friction fingertip material to interact with the item. This increases the gripper’s grip on the products with complex geometry and can reduce the pressure that the gripper needs to apply in order to pick up the product. A second solution could be using a ball in socket joint between the gripper finger and fingertip. This allows much more flexibility for the fingertip to adjust its pitch around to directions.

These may make the gripper slightly more complex, which is why the Adaptive Robot Gripper team did not have time to implement these improvements, but they should increase the ability for the gripper to pick up more items, especially those whose containers have complex shapes. This is especially important for container designs of cosmetic and hygiene products.

Generative Design to Increase Payload

The final version of the Adaptive Robot Gripper was made of Aluminum. While this version of the gripper was sturdy and heavy duty, the weight of the aluminum gripper was much more than that of the plastic gripper. To improve this, the Adaptive Robot Gripper team would suggest implementing a Generative Design. This means a design the uses metal truss like structures to remove the amount of material used while maintaining the strength of the solid material. Problems with this improvement is that it would be expensive and the team continuing this project would need the advice of someone who would know how to implement a regenerative design. However, if this was able to be implemented, it would give the gripper the ability to increase its payload because the weight of the gripper was reduced.

This design was one suggested by Unilever when an aluminum gripper was about to be created. The reason that it was not implemented in the first iteration of this project was that there was no team member that head experience with this or knew anyone who had experience with this form of material design. In addition, there was little time to attempt a regenerative design, but it could be an improvement done by a team continuing this project.
The Next Steps section will discuss the actions that a team, continuing this project, could take to eventually implement the Adaptive Robot Gripper into a Unilever manufacturing line. The items discussed in this section will be in the order that they should incorporated.

1) Fix Some of the Mentioned Improvements

The more important improvements such as the tolerancing and the PCB upgrade would be best to accomplish first. These improvements should be done to improve the overall improvements of the system. The force sensor improvement should also be experimented with to attempt to reintegrate and redesign the feedback loop so the gripper could more accurately control the force applied to the products. The fingertip and generative design improvements can be considered after the Adaptive Robot Gripper has been integrated with a robot arm because these are more improvements to increase the number of products that the gripper can pick up. These improvements are more expensive than the rest and would take a lot of time away from integrating the system with a robot arm.

2) Interface with Industrial Robot Arm

Since the purpose of this project is to produce a robot gripper that interfaces with robot arms on Unilever’s manufacturing lines, the first step outside of improving the current system should be to start incorporating a robot arm into the system. To accomplish this, a physical robot arm and training for would need to be provided early in the project so that the team has time to work with the robot arm and learn how they can control it with the current Adaptive Robot Gripper system. This is something that Unilever attempted with the first team that worked on the Adaptive Robot Gripper project, but there was not enough time to add this component into the system.

To interface with the current system of the Adaptive Robot Gripper, they would need to first make a wrist attachment for the gripper to be secured to the robot arm. Then they would need to see what hardware they would need to connect the robot arm to the Raspberry Pi controller. The RoboDK can control the robot arm, so the team would need to install the RoboDK software onto the Raspberry Pi and contact the RoboDK company to extend the license to the Raspberry Pi. With the communication that was made with RoboDK, they confirmed that they could allow a license to cover a personal laptop and a Raspberry Pi for this project. They would then need to figure out how the Raspberry Pi can communicate with the robot arm, whether that is through cables or through wireless means. Once this is set up, the RoboDK software should have the ability to use python code and convert it to code specific to the robot arm to control it.

This step is important to show that the system can interface with robot arms to accomplish the goal of picking up Unilever product by being able to properly orient the arm and grabbing the products in the proper location.
3) Work with Conveyor Belts

Once the robot arm has been integrated into the system, the next step would be to have a conveyor bring items to it. When a conveyor brings a product in front of the robot arm, the robot arm should be able to time the conveyor properly to track the product and pick up the item. To simplify this at first, the conveyor belt should be a conveyor that starts and stops at consistent intervals. The team would need to figure out how to track the products on the conveyor through timing, and then set that time into the controlling raspberry Pi so that the system has the ability to know where a product is on the conveyor after the system identifies the product. The system already can find the horizontal distance of a product on the conveyor, so the timing would provide the distance along the axis in the direction of the movement along the conveyor. This is not something that needs to be added, but it is a slight adjustment that needs to be made to the current software.

The team would need to set a requirement, that Unilever agrees to, to set a goal to decide performance success for the first conveyor before moving on to work with a conveyor that moves at a constant speed. Then, when the whole system works well with that style of conveyor, the team should implement a constant speed conveyor. This is harder because the robot must track the moving product while it is in the process of picking it up. The system would still be reliant on the timing to understand where the product is on the conveyor, but it does not have the advantage of a stopped product when it is in the process of picking the product up. Implementing a conveyor shows that the software and the robot arm has the ability to track products as they move down conveyors.

4) More Product Variety

Once all of the other steps have been accomplished, then teams continuing this project can train the system to include more products. Since Unilever has many products under its brand, there are plenty of product that this system can be trained to work with. This should also be the easiest step because the system already can be trained for other products, but there was not time to add more, and it is more important to have the system completely working and able to be implemented into a manufacturing line before training the system to work with more products.

This is also the step where the fingertip and generative design should be considered if the project has extra budget and time to spare on these improvements. This would further increase the amount and types of Unilever products that the gripper and robot arm can pick up.
Summary
The Adaptive Robot Gripper System sponsored by Unilever UK demonstrates the ability to detect a product on a manufacturing line. The Team for this year consists of 5 seniors in several different engineering majors including Biomedical, Mechanical, Electrical, and Optical Engineering.

The system then images the product and identifies it through the use of YOLOv5 image recognition model. Once identified the product moves to the gripper before being picked up and moved. The Adaptive Robot Gripper System has been carefully designed, built, and tested to the level of functionality as required by Unilever UK. The project ends development at the end of the academic year with a physical system that demonstrates the goal of an eventual gripper system to be deployed at a factory. This includes two raspberry pi’s, several prototype grippers both plastic and metal, a SONAR 1D sensor, a raspberry pi camera, a conveyor belt, and a tv for demonstration purposes.

The Adaptive Robot Gripper was developed with improving efficiency of manufacturing lines by reducing time to change types or specifications of grippers on the lines. This lost productivity accounts for significant impact on the finances of the manufacturing lines. The system as was developed as a learning tool for relevant industry skills such as python, project management, communication, 3D printing, and much more.

At this time, we would like to thank Unilever UK for sponsoring this project and their constant commutation and help from the other side of the world throughout this project. Special thanks to Dr. Slepian for connecting with Unilever UK, Don McDonald our professor for his support of our project outside of class periods, Dr. Enikov for opening lab space for us and major contributions to the motor control portion of our project.
Appendix

ASME Y14.5 and Y14.100 were used as guidelines for the gripper mechanical drawings.

Figure.1 Palm Part

Figure.2 Link Part
Figure 3. Gear Link

Figure 4. Finger Part
Figure 5. Grip Part

Figure 6. Drive Gear Part
Figure 8. The assembly of the adaptive robot gripper
Figure 9. The assembly of the adaptive robot gripper (exploded view)

<table>
<thead>
<tr>
<th>ITEM NO.</th>
<th>PART NUMBER</th>
<th>QTY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PalmPart</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Gear1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>LinkPart</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>FingerPart</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>DriveGear</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>GRIP</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 10. Yaskawa GP8 adapter
Adaptive Robot Gripper

RoboDK User Manual

Version: 1

Team: Lucas Hawley, Connor Nagore, Joe Liang, Riyadh Alswayel, Max Critchfield

Sponsor: Unilever
Table of Content

Introduction and Scope.................................................................................................................................55
Defining Areas of the RoboDk Displays..........................................................................................................56
Main Menu.......................................................................................................................................................56
Main Display....................................................................................................................................................56
Station Tree....................................................................................................................................................56
Status Bar.......................................................................................................................................................57
License and Software Download Instructions.................................................................................................57
  -Inserting License Code.................................................................................................................................57
Training............................................................................................................................................................58
General............................................................................................................................................................58
New Station.....................................................................................................................................................58
Renaming.........................................................................................................................................................58
Selecting Multiple Items in the Station Tree.....................................................................................................59
Selecting a Group of Items in the Station Tree.................................................................................................59
Renaming a Group..........................................................................................................................................60
Moving perspective of station..........................................................................................................................60
  -Translation..................................................................................................................................................60
  -Zoom............................................................................................................................................................61
  -Rotation.....................................................................................................................................................61
Inserting Objects into Station..........................................................................................................................61
  -Local Files................................................................................................................................................61
  -RoboDk Online Library..............................................................................................................................62
  -Inserting Robot..........................................................................................................................................64
  -Local Files Solid Works............................................................................................................................64
Adding Reference Frames...............................................................................................................................66
Moving an Existing Reference Frame...............................................................................................................67
  -Move with the “alt” Key..............................................................................................................................67
  -Move with the Move Reference Frame Icon.............................................................................................67
  -Move with Frame Details..........................................................................................................................67
Adding Targets.........................................................................................................................................................................................68
Moving an Existing Target.....................................................................................................................................................................................68
- Teach the Target a New Position........................................................................................................................................................................68
- Move with the “alt” Key..................................................................................................................................................................................69
- Move with the Move Reference Frame Icon........................................................................................................................................69
- Move with the Target Window.................................................................................................................................................................69
Positioning and Orientation.............................................................................................................................................................................70
Moving the Robot Arm.....................................................................................................................................................................................71
- Move with the “alt” Key................................................................................................................................................................................71
- Move with the Move Reference Frame Icon........................................................................................................................................72
- Move with Robot Panel.............................................................................................................................................................................72
- Changing Speed Settings of Mechanisms...............................................................................................................................................74
Programs ........................................................................................................................................................................................................76
- Joint Movement.........................................................................................................................................................................................76
- Linear Movement....................................................................................................................................................................................76
- Circular Movement..................................................................................................................................................................................77
- Add Event........................................................................................................................................................................................................77
  -- Attach Object.......................................................................................................................................................................................77
  -- Detach Object....................................................................................................................................................................................78
  -- Changing the Visibility of Items.......................................................................................................................................................79
  -- Show Object/Tool.............................................................................................................................................................................79
  -- Hide Object.......................................................................................................................................................................................80
  -- Set Absolute Object Position.......................................................................................................................................................80
  -- Set Relative Object Position.......................................................................................................................................................81
  -- Init........................................................................................................................................................................................................82
  -- Changing the Settings of an Existing Event....................................................................................................................................83
- Pause Tool.......................................................................................................................................................................................................83
- Inserting a Python Script..........................................................................................................................................................................83
- Program Call/Insert Program Instructions.........................................................................................................................................84
- Add Instruction.........................................................................................................................................................................................85
- Program Loop.........................................................................................................................................................................................86
- Speed Settings.......................................................................................................................................................................................86
  -- Speed Up the Simulation................................................................................................................................................................86
- Progress Bar.......................................................................................................................................................................................................88
Introduction and Scope

RoboDk software was used for the emulation aspect of the Adaptive Robot Gripper project. The main purpose of this emulation software was to take the place of a physical robot arm to show the movements of the arm once the adaptive robot gripper software recognized a product on the conveyor belt. This software was able to simulate the set-up and function of the Adaptive Robot Gripper project along with showing how the system would interact with the chosen products and conveyor belts in a manufacturing environment. In addition, this software has the ability to produce code to control the robot arm, that was chosen by the user in RoboDK, in the specific language that the robot is controlled by. Given this, the RoboDK software can be downloaded onto the Raspberry Pi and coded with the Raspberry Pi that controls the Adaptive Robot Gripper system because the RoboDk program runs off of python like the rest of the Adaptive Robot Gripper system. Programs in the robot language can then be created off of the python programs made in the system. This gives the controlling Raspberry Pi the potential to be connected to the robot arm to control it along with the rest of the Adaptive Robot Gripper system. Note that interfacing and controlling a physical robot arm is outside of the scope of this project; however, it is important to note the potential of the software chosen for emulation in the Adaptive Robot Gripper project.

The purpose of this manual is to walk a user through the tools that were used in the creation of the Adaptive Robot Gripper project simulations. While RoboDk has many more functions then will be presented in this manual, this document will cover all of the tools necessary to create a simulation that closely simulates the manufacturing environment that the adaptive robot gripper will be placed in.

For the adaptive robot Gripper project, Unilever requested the use of the Universal Robot UR-10e for the modeled robot arm I the software. Since this was the model used in the project, it will be the example robot arm throughout this manual. This software also allows the user to import models for tools and objects from Solid Works. For this project, the Adaptive Robot Gripper team was able to import Solid Works models the Unilever products that were gripped in this project along with the model of the gripper created in this project. When discussing the importing of Solid Works tools and objects, the model of the gripper will be the example used in this manual.
Defining Areas of the RoboDk Displays

This section defines different areas of the RoboDK station display so that these areas may be talked about without confusing the reader. Under this section, the image below can be referenced to visually separate the display into areas defined by their specific labels.

Main Menu

The main menu is where a user can find general commands for the software. Outside of the typical commands to save, this is not a part of the display that is used often. However, it is still valuable to define the space to separate it from the other parts of the RoboDK station display.

Main Display

The main display is the part of the station where the user interacts most with the robot, tools, and objects inputted into the station. This area covers the rest of the software display not defined by another label.

Station Tree

This part of the display organizes and groups stations, frames, and programs together. Multiple stations can be shown in this part of the display, and they can be expanded to show their content of reference frames and programs. The content within these stations can continue to expand to show which reference frame each object in the station falls under. Programs can also be expanded to show the series of commands that are included within them.
**Status Bar**

The status bar shows the progress of a simulation in reference to the total amount of time that it takes to run a whole program. This information is displayed when a program is in progress and is left empty until a program is clicked on.

**License and Software Download Instructions**

To download the RoboDK software onto a computer or laptop, the user can go to [https://robodk.com/download](https://robodk.com/download) where they can find the latest versions of the RoboDK simulation software to download.

To obtain the full version of the software on a user’s computer or laptop, the user needs to purchase a license from RoboDk. License purchase options can be found at [https://robodk.com/pricing](https://robodk.com/pricing). Here they offer commercial and education version of the software license. For the purposes and budget of the Adaptive Robot Gripper project, the education license was purchased and used. The license will allow the user to use every tool within the software. While the education license was slightly limited compared to the commercial version, there were no obstacles during the use of this software for the Adaptive Robt Gripper Project. Note that an education license is only good for a single device. Since the commercial license was out of the price range and not necessary to the Adaptive Robot Gripper project, the limit of devices for a commercial license is not known. A free version is also available and is the default software immediately after download.

**Inserting License Code**

Once a license has been purchased from RoboDk, the user will be given a license code. The user needs to go to the Help Option in the Main Menu and choose the License option in the drop-down menu.

![License Options Window](image)

The License Options window will open, and the user needs to then choose the Network option. There, they will be given the option to enter a Server Code.
This is where the user must enter their license number given to them after purchase.

**Training**

Free online training is offered through Youtube.com where RoboDK has a channel. There, instructors walk the user through basic functions on the program. For more advance options, users have the option of signing up to RoboDK forums to ask questions to a community of experienced RoboDK users. This forum can be found at [https://robodk.com/forum/](https://robodk.com/forum/).

**General**

**New Station**

After opening the software, pressing the “Ctrl” and “n” keys together result in the creation of a new blank station.

**Renaming**

This process of renaming works for any file, object, reference frame, target, or program displayed in the Station Tree. To rename an item, click on the specific item so that it is highlighted. The first method of renaming an item is to press the “F2” key and this allows the name to be changed.
The second method to rename an item in the Station Tree is to click on the item so that it is highlighted, then right click on the item to open a pop-up menu. Find the “Rename” tool in the pop-up bar and click it which will allow the item to be renamed. The second method cannot rename stations.

Selecting Multiple Items in the Station Tree

RoboDK offers an easy way to select more than one item in the Station Tree. By holding the “Ctrl” key, a user can highlight multiple objects with their cursor in the Station Tree.

Selecting a Group of Items in the Station Tree

A user may need to select multiple items that are all right next to each other. Instead of holding the “Ctrl” key and selecting each item individually, a user can select the top or bottom item in a group, then press and hold the “Shift” key, and choose the item at the opposite end of the group. This will select the entire group in the Station Tree.
Renaming a Group

If a whole group is together and in the order that the user desires, then RoboDK offers the option to name a group of items the same while giving them different numbers. Using the selecting multiple items or selecting a group of items, as seen earlier in this manual, the user can then right click one of the highlighted items which then opens up a pop-up menu. To rename the group of items chosen by the user, they need to choose the Rename Group option.

This option will give the group the same name that the user enters into the Rename window.

All of the highlighted options will be given the name that the user placed in the Rename window and they will be numbered according to the order that they were listed from the top of the Station Tree.

Moving perspective of station

-Translation

To move through translation, hold the “Crtl” and “Alt” keys and then hold the left click of the mouse or touch pad. The view of the station can be moved up, down, right, and left.
-Zoom

There are two methods to zoom into any part of the station. The first is using the scroll on a mouse or laptop touch pad. The second method is to hold the “shift” key while also holding the left click of the mouse or touch pad. Moving the cursor up and down controls the zoom in or the zoom out.

-Rotation

To rotate the view of the station, the user needs to hold the “Shift” and “Ctrl” keys while holding the left click of the mouse or touch pad. Moving the cursor up, down, right, or left controls the rotation of the station view.

Inserting Objects into Station

There are two ways in which objects can be brought into the RoboDK station. Through local files and online. This section will explain how to achieve objects through each method.

-Local Files

Once the RoboDK software has been downloaded onto the user’s computer, some files come with the program that the user can quickly access without needing to go through the internet. To start the search for the local files, click on the Load Files (Local) tool in the station toolbox.

Clicking this tool brings the user directly to the RoboDK folder that came with the software download.
In this folder, various example stations, scrips, robot models, tools, and objects can be accessed directly through this method. To import any of these items, the user just needs to select the desired object from this folder window, and it will be brought into the current station. Example stations are marked in the folder with a RoboDK logo, example scripts are marked with a logo depicting a paper and the python logo, robot models are marked with an image of an orange robot, tools are marked with a gripper logo, and objects are marked with a cube.

While there is a lot of content that comes in this local folder, it may be missing some things that the user is looking for. There are a few other ways of inserting items that cannot be found in the local files.

**-RoboDk Online Library**

If the user cannot find a desired item in the local RoboDK files, then they can access the RoboDK internet library to find many more items to include in their station. This can be accessed through the Open Online Library Icon in the station toolbox.

Once this icon is clicked, it will open the user’s main internet browser and take the user to the website for RoboDK’s online library.

When using the library to look for a specific item, the filter box will be a very useful tool for quick browsing. The brand filter is specifically for Robot brands which holds many different brands of
robot arms that the user can upload to their station. Type is another useful filter that allows the user to only see specific item categories such as robot, objects, and tools.

There is a lot more content in the library and filters that can be added, but in this manual, the most basic and useful ones were covered.

When the user wants to pick an item, they need to hover their cursor over the item that they want. The image of the item is replaced with a blue rectangle that shows four words: “Open”, “View”, “Download”, and “3D View”.

To upload the desired item to the RoboDK station, the user must click the “Download” option with will make the browser download window open.

Once the user chooses the open option on this window, the item will be uploaded directly to the RoboDK station where the Open Online Library Icon was originally pressed.
-Inserting Robot

The previous two sections (Local Files and the RoboDK Online Library sections) described how to find items in the local RoboDK files on the user’s computer or finding an item through RoboDK’s online library are both valid ways of uploading the user’s desired robot to their RoboDK station. To find a more specific robot, the online option is better due to the number of options in the online library.

To find a robot in the local RoboDK files, the user must look through the options that are marked with an orange robot logo. Each robot is specifically named so there should be no confusion about the identity of the robot.

To find a robot through the online library, the user must know some specific details about the desired object such as the brand and name of the robot. It is possible that it is not yet included in the RoboDK online library; however, RoboDK claims that anyone can contact them and ask for robot models to be added to their library. Otherwise, using the brand filter should narrow the search so that the user can easily find the robot that they want.

The robot should be the first item that the user uploads to their station. The one acceptation being a table or something that the user may want underneath the robot arm in their station.

-Local Files Solid Works

The user may have tools, products, or other environment obstacles or objects that the user may want to include in the RoboDK station. One of the ways in which the user can insert a unique object is through modeling that item in Solid Works and then importing it into RoboDK. If the user desires to bring in an object from Solid Works, it is possible to import this item through the local files method. First, the desired tool or object needs to be created in Solid Works. Once complete, the user must click Save As in the File tab of main menu in the Solid Works Program with the desired part open.
The user must choose the desired file destination of this object on their computer and name it as they so choose. Whether this item from Solid Works is a tool or object, it should be saved as a “STEP AP214” file. As shown by the cube in the file window, this will save the tool or object as if it were an object in the RoboDk local files.

After being saved, open the RoboDk station that the tool or object is to be uploaded to. Click on the Load Files (Local) tool in the station toolbox as if the item could be found in the local RoboDK Files. Once the file window opens, the user needs to use the window to browse for the location that the tool or object was placed. Once that location has been opened in the file window, click the item that is to be imported into the station. No matter the type of item, whether it be a tool or object, this will upload it to the main station into the main frame of the station as an object. As shown below, this is the same frame as the base of the robot arm.

If the item is an object, RoboDk will initially treat it properly by marking it with a cube logo. Highlighting the item by clicking it in the Station Tree allows the user to move it to different reference frames. Doing so changes the reference frame by which its position is defined. Moving the objects position with reference to its reference frame will be covered in a later section titled Positioning and Orientation.

If the item were a tool that is intended to be connected to the robot arm at the flange, then the initial assumption made by RoboDk to mark the tool as an object is wrong. To correct this, highlight the tool by clicking and holding the item in the Station Tree. The user must hold the object over the robot symbol and let it go once a plus sign appears next to the mouse.
This moves the tool to the flange of the robot. If the tool is not oriented properly, go to the Positioning and Orientation section of this documentation to see an explanation for changing the orientation of the tool with respect to the reference frame of the robot arm flange.

Adding Reference Frames

A reference frame can define a position of an object with respect to the main reference frame of the station. New reference frames can be added to the RoboDK station by clicking the Add Reference Frame icon.

When a new reference frame is added to a station, its location and orientation is based off of the main station reference frame. In the Station Tree, it is not defined under any other reference frames in the Station.

These new reference frames can be dragged into other reference frames in the Station Tree to define the new reference frame under another one. This is done by left clicking the name of the new reference frame in the Station Tree, holding the name of the reference frame and moving it with a cursor until it is on top of the reference frame that the user wants it defined under. Once a plus sign appears next to the cursor, release the new reference frame and it will now be defined under the desired reference frame.
Moving an Existing Reference Frame

-Move with the “alt” Key

To move any reference frame, the user must start by hold the “alt” key. Doing so should make arced arrows and boxes appear around the target reference frame.

The squares move the reference frame along the planes, or the two directions of the arrows that the square is shown between. The arced arrows rotate the target reference frame around the three different directions. In order to manipulate the arced arrows rotate the target reference frame around the three different directions. In order to manipulate the object in any of these directions and orientations, the user must continue to hold the “alt” key while holding the left click on the mouse or keyboard. While holding both of these buttons, the reference frame can be moved in the chosen direction by moving the cursor to the location or orientation that the user desires.

-Move with the Move Reference Frame Icon

Rather than holding the “alt” key, the Move Reference Frame icon can be used to display the squares and arced arrows around the target reference frame to manipulate the location of any reference frame.

-Move with Frame Details

The second method of moving a reference frame can be done by highlighting the name of the reference frame in the Station Tree by left clicking it. Then the user must press the “F3” key to open the Frame Details window for the chosen reference frame.
The location and orientation of the target reference frame can be controlled under the Reference Position section of the Frame Details window. This defines the location and orientation of the reference frame with respect to the main reference frame that it is defined under. This window can control location of the reference frame in the x, y, and z directions of the reference frame it is defined under, along with rotation about each of these directions. The red box defines x direction positioning, the green defines the y direction positioning, the dark blue box shows the z direction positioning, the light blue box defines rotation about the x direction, the pink box defines the rotation around the y direction, and the yellow box defines the rotation about the z direction. If the user starts in the far-left box, the x direction manipulation box, then they can use the “tab” key to toggle right to the other direction manipulation boxes. The definition of the location boxes is the same for the Target window, Tool Details window, and Robot Panel which are talked about in later sections.

Adding Targets

A target is a location for the reference frame of the robot flange to later return to. The robot flange is the part of the robot that a tool can be attached to. A target can be defined into the RoboDK station by first positioning the robot into the user’s desired location and orientation for the target to be defined by. Movement and orientation controls for the robot will be covered by a later section titled “Moving the Robot Arm”. Once in the desired location, the user must click the Add Target icon in the Station Tool Bar.

Moving an Existing Target

-Teach the Target a New Position

If the user wants to change the position of a defined target, then there are two methods of moving the position of the target. The first of which being the user can move the robot to the position that they chose the target to move to. Once in position, then the user can right click on the name of the target they want to move in the Station Tree. Doing so will make a pop-up window appear where the user must choose the Teach Current Position option. This will move the target to the new desired location.
-Move with the “alt” Key

The second way in which the target can be moved is by clicking and holding the “alt” key. Doing so should make arced arrows and boxes appear around the target reference frame.

The squares move the target along the planes, or the two directions of the arrows that the square is shown between. The arced arrows rotate the target reference frame around the three different directions. In order to manipulate the object in any of these directions and orientations, the user must continue to hold the “alt” key while holding the left click on the mouse or keyboard. While holding both of these buttons, the target can be moved in the chosen direction by moving the cursor to the location or orientation that the user desires.

-Move with the Move Reference Frame Icon

Rather than holding the “alt” key, the Move Reference Frame icon can be used to display the squares and arced arrows around the target reference frame to manipulate the location of any target.

-Move with the Target Window

The final way that a target can be moved is by highlighting the name of the target in the Station Tree by left clicking it. Then the user must press the “F3” key to open the Target window for the chosen target.
Under target type, there are two options for the target: keep cartesian position and keep joint values. The first option just has the robot flange to return to the target location with the same orientation of the flange reference frame. The keep joint values option constrains the robot to return to the same target location while maintaining the configuration of each joint of the robot arm as defined by the user.

The location and orientation of the target reference frame can be controlled under the Target Position section of the Target window. This defines the location and orientation of the target reference frame with respect to the reference frame that the target is defined under. This section of the window can control the location of the target in the x, y, and z directions of the reference frame it is defined under, along with rotation about each of these directions. Controls in this section of the Target window are similar to that of the Frame Details window.

Defining the configuration of the robot arm will be covered in the later section titled “Moving the Robot Arm”. This covers the bottom section of this window that shows the orange robot logo and shows values for the robot joints.

**Positioning and Orientation**

Once a tool or object has been dragged into the desired reference plane, the user may want to change the orientation of the item. This can be done by highlighting the item in the Station Tree by clicking the name of the item or clicking on the item directly so that the item itself is highlighted with a slight blue tint. Once either of these methods are performed, the user can then press the “F3” key on the keyboard which opens the Tool Detail window for that specific object. The Tool Details can also be opened by double clicking on the item directly.
In this window, the user can adjust the orientation or positioning of the item’s reference frame with respect to the reference frame that the user attached the item to. As shown by the image, the item can be move in the x direction, y direction, and z direction with respect to the reference frame defined by the user. This window can also rotate around the 3 directions previously mentioned. The directions of the colored boxes are covered under the Target window description under the Adding Targets section. The user can start with the x direction, the red box, and to move through the directions and orientations, the user can click the “Tab” key on their keyboard adjust these options more quickly and conveniently.

By clicking on the more options button, the Tool Details window give the user the option to change the orientation of the item with respect to its own reference plane through the Move geometry section of the Tool Details window.

The item can be moved and rotated in the same way as previously mentioned, but with respect to the item’s own reference plane. The units of measurement used in the RoboDK station is in millimeters. This is convenient to know because the user can use measurements from their actual environment and accurately simulate them in the RoboDK.

**Moving the Robot Arm**

- **Move with the “alt” Key**

  Once the user has chosen a robot and imported it into their RoboDK station, it can be moved in a few different ways. The first method of movement is by using keyboard keys and the cursor. By holding the “alt” key, squares appear around the movable reference frames.
This allows the user to grab a square and move the robot around on a plane, or in two different directions when holding a square with the cursor. Along with the squares, arced arrows appear control the rotational movement around the axis of the reference frame that the user is manipulating. This method works for moving reference frames regarding the robot that are visible. This means that the reference frames that can be moved with this method are reference frames regarding the tools center point (TCP), robot flange, and for targets.

-Move with the Move Reference Frame Icon

Another way to move the robot is to select the Move Reference Frame Icon found in the Station Toolbar.

Selecting this icon shows the squares and arced arrow around the robot flange reference frame as if the “alt” key was being held. This allows the user to manipulate the robot arm with their cursor.

-Move with Robot Panel

The final way to manipulate it is to double click on the robot to open the Robot Panel window. This panel can also be opened by clicking on the robot’s name in the Station Tree and then pressing the “F3” key. This window is similar to the Tool Details window to move the tools and objects; however, there are some additional controls specifically for the robot.
The first section has three rows of colored boxes similar to the Frame Details window, the Target Window, and the Tool Details window. The directions of the colored boxes are covered under the Target window description under the Moving Reference Frames part of the Adding Reference Frame section. The difference with this window is that the reference frame of the base and tool can be adjusted with respect to the main reference frame of the station. The top reference frame adjustment moves the reference frame of the tool with respect to the reference frame of the robot arm flange.

The second part of the Robot Panel are additional controls for movement of the robot. Any of the movable reference frames of the robot can be chosen and the specific direction of translation or rotation can be chosen to be adjusted by the nob. This portion of the window also allows the user to see the range of the tool, robot flange, and robot wrist center. The final controls in this part of the window control the visibility of the tool and robot arm reference frames.

The third part of the window adjusts the angle of each joint. The top slider represents the orientation of the base joint, and the bottom slider represents the flange orientation. As the user moves down the list of sliders, they are moving up the robot arm starting at the base joint and moving towards the robot flange joint. The orientation can be adjusted precisely by entering angle values or more generally by adjusting the slider.

The third part of this panel also has a Home button. Selecting this button takes the robot back to a default orientation. While this note is not necessary, the user can make a target at this location for the robot to have a default, or home orientation, when moving through a series of targets in a program.

The Align button is useful when a user is attempting to make the tool perpendicular to a plain after manually adjusting the joints. Choosing this option will automatically move the tool and robot arm to a position where is it more perpendicular to the plane which the user has moved it near perpendicular to.
The final section of the Robot Panel window shows the Other Configurations option. This drop-down menu suggests the best orientations for the whole robot arm to reach the current location and orientation of the reference frame of the tool and robot flange reference frame.

Selecting the More Options button allows the user to be more specific in finding orientations that the robot to assume for reaching its current location and orientation of the tool reference frame and the robot flange reference frame.

The More Options window for the Robot Configurations allows the user to choose filters for the specific orientation that is desired. The user can then go through the suggested options and find the one they feel best suits the environment for their station.

Once these settings have been chosen, the user can click the “OK” button and the configuration will be set up.

- Changing Speed Settings of Mechanisms

When industrial and manufacturing settings are considered, it is preferable to increase the speed and acceleration at which the robots, conveyor belts, and other mechanisms perform. The RoboDK software offers the option to increase or decrease the normal speed at which the mechanisms in the simulation perform, and this can be done for each individual mechanism. For this explanation, the UR-10e Robot Arm will be used as an example. However, the method for changing the speed and acceleration of the mechanism is same for all robot arms and mechanisms, such as conveyor belts, in the station.

To adjust the speed and acceleration of a mechanism, the user must first open the mechanism panel for the desired mechanism that they want to adjust. The process for opening the mechanism panel is explained in the Move with Robot Panel in the Positioning and Orientation section of this manual. Once opened, the user needs to
The user then needs to go to the parameters button in the top right corner of the panel next to the name of the mechanism. This opens the Robot Parameters window.

This window is the same whether the user is adjusting a robot arm or conveyor belt. It offers more advanced settings for the mechanism’s movement.

The main concern for this section is the Speed (mm/s), Acceleration (mm/s²), Joint Speed (deg/s), and Joint Acceleration (deg/s²) options at the top of the window. These settings allow the user to enter an exact speed and acceleration that the user wants the mechanism to move. It also allows the user to adjust the angular acceleration of the joints of the mechanisms.

These speed settings will be experience when a program is chosen, and the simulation is playing. Creating and simulating programs will be explained under the Programs section of this manual.
Programs

Programs are a series of instructions that the station executes in order to simulate robot operation. It is best practice to make a series of smaller programs that can later be called in larger programs. In the long run, this simplifies the program because it allows the user to repeatedly use a smaller program instead of remaking it multiple times in larger ones.

Before a program is created, it is best to have all of the desired targets defined. The add program tool in the Station Toolbar adds an empty program into the Station Tree.

Programs are made up of instructions which include robot movement, pauses, and events that are placed in a desired order by the user. To place a movement into the program, the user must hold the “Ctrl” key and select the desired target and the specific program that they want the target to go into. Once the target and program have been chosen, the “Ctrl” key can be released and the user must choose one of the three movement options for the robot arm: Joint Movement, Linear Movement, and Circular Movement. This places a robot arm movement into the program. Repeating these steps for the same program adds another robot arm movement under the one previously added. If the user decides to change the order of the movements in the program, then they must left click and hold the movement that they want to move and drag it to the part of the series that they desire the movement to occur. It can be held above or below another movement and dropped which will place it where the user has chosen. Programs are not limited to movements, but can also include events, pauses, and python scripts. All of these will be covered in this section.

Once a program has been created with all the events and program calls that the user requires, they only need to double left-click the specific program that they want to run and the simulation that was chosen will be performed by the station. Adding events and program calls will be discussed in the next series of subsections.

-Joint Movement

A joint movement is chosen when the user wants the robot to move to a specific target and end up in the orientation defined by the user when the target was previously created. The Joint Movement icon logo shows two targets with a curved line.

-Linear Movement

A linear movement is chosen when the user wants the robot arm to move to a chosen target; however, the orientation of the robot arm at that target is not important. The robot flange and tool will be in the position and orientation defined when the target was originally created, but the rest of the arm will be configured as the software figures to be best. This will result in the tool and robot flange to
appear to move in a linear path from the position before the movement was activated to the target defined by the user. The Linear Movement icon shows two targets connected with a straight arrow.

- **Circular Movement**

The circular movement tool is slightly different from the other two movement options. When holding the “Ctrl” key, two targets must be chosen along with the program that the user wants the circular movement to be a part of. This creates an arc path between the initial location that the robot arm was previously set at and the two targets. The order that the robot arm moves through these two targets are in the same order which the user clicked the targets to be a part of the movement. If these two targets are on a curved surface or obstacle to the robot arm, the circular path will be curved along the surface of the obstacle. The Circular Movement icon shows three targets connected by a curved arrow.

- **Add Event**

The add event tool has more to do with the simulation of the environment around the robot arm rather than the direct movement of the robot arm. To add an event to the program, click the Add Event icon in the Station Toolbar and the Event Instruction window will pop up.

-- **Attach Object**

The attach object option is found in the far-left drop-down menu of the Event Instruction window.

Choosing this option will allow the user to select the tool that the item will attach to in the top drop-down window to the far-right side of the Event Instruction window. When this event is called, the object that is near the gripper will stop respecting the previous reference frame that it was defined
under, and it will be defined under the tool chosen by the user. This is how the software simulates items being attached to the tool on the robot arm.

The second drop down menu on the far-right side of the window allows the user to change the settings for the distance range that the item will attach itself to the chosen tool. For the adaptive robot gripper project, this setting was left as default. The user must make a program moves the robot arm into position to simulate the system picking up a product where the gripper tool center point is at the calculated distance from the product with the desired orientation. Leaving the “Measure Distance as:” setting at default attaches the item to the gripper, in the orientation set up by the user when the event is called, as long the reference frame falls within the maximum distance showed by the “Maximum distance (mm):” setting in the Event Instruction window. There is no need to set a specific object for this option because the RoboDK software will choose the closet item to the Tool Center Point of the tool chosen by the user.

Once these settings have been chosen, the user can click the “OK” button and the event will be set up.

--Detach Object

The detach object setting can be called in a program after an item has been attached to the tool chosen by the user in an attach object event. This option can be found in the action menu which is the drop-down menu on the far-left side of the Event Instruction window.

On the far-right side of the Event Instruction window, the first drop-down menu allows the user to choose the tool that the object is released from. The user should choose the same tool that was chosen in the attach object event.

The second drop-down window on the far-right side of the Event Instruction window is the “Attach to Parent:” setting. This setting allows the user to choose the reference frame that the object is attached to once the object has been released from the chosen tool. In the adaptive robot gripper project, the reference frame chosen was the movable reference frame on the conveyor belt. Conveyors will be talked about more in the Conveyor section of this manual.

Once these settings have been chosen, the user can click the “OK” button and the event will be set up.
--Changing the Visibility of Items

Making an object visible or invisible can enhance a simulation by only showing the items that the user wants to show and hiding any items or reference frames that the user may not want their audience to see in the final product. During a simulation, the Event Instructions allow the user to show or hide any object with an instruction that the user puts into a program. The next two sections will talk about how to insert an instruction into a program to show an item that was previously invisible or to hide an item that was previously visible. This section will give a quick description of how a user can change the visibility of any item or reference frames.

RoboDK offers the option to show or hide any item, reference frame, or target. This can be done by right clicking the object in the station or the object’s name in the Station Tree and selecting the visible option in the drop-down menu.

For tools, objects, and robot arms, this drop-down menu offers the user to change the visibility of the item itself, and the reference frames for that specific object.

--Show Object/Tool

The show object/tool option in the Event Instruction window allows the user to insert an event into a program that shows a previously invisible item when the event is called by the program. This option can be chosen in the drop-down menu under the “Action:” setting on the far-left side of the Event Instruction window.
Choosing this option gives the user the option to pick which item or reference frame they would like to make visible with this event. The user can select the desired item in the “Select objects/tools to show:” section of the Event Instruction window. Once these settings have been chosen, the user can click the “OK” button and the event will be set up.

--Hide Object

The hide object/tool option in the Event Instruction window allows the user to insert an event into a program that hides a previously visible item when the event is called by the program. This option can be chosen in the drop-down menu under the “Action:” setting on the far-left side of the Event Instruction window.

Choosing this option gives the user the option to pick which item or reference frame they would like to make visible with this event. The user can select the desired item in the “Select objects/tools to hide:” section of the Event Instruction window. Once these settings have been chosen, the user can click the “OK” button and the event will be set up.

--Set Absolute Object Position

The Set Object Position (Absolute) option in the Event Instruction window allows the user to insert an event into a program that will return an item or reference frame back to a location that was chosen when this event was created. The location will be set in an absolute position with respect to the
main reference frame of the station. This option can be chosen in the drop-down menu under the “Action:” setting on the far-left side of the Event Instruction window.

Once this option is chosen, a user can then choose the item that they want to save the current position under the “Select objects to save current position (absolute)” part of the Event Instruction window.

When a program is run with an absolute position, if the object has been moved, then it will return to the saved location when the event is called. The saved location is the location that the item was at when the event was created by the user.

Once these settings have been chosen, the user can click the “OK” button and the event will be set up.

The location of the object can be updated by modifying the event, which will open the Event Instruction window with the settings that were previously set. The user has to re-select the object in the “Select objects to save current position (absolute)” part of the Event Instruction window and then click the “OK” button. The new location will be saved into the event once this has been completed. The modification of the events will be covered by the “Changing the Settings of an Existing Event” section later in the manual.

--Set Relative Object Position

The Set Object Position (relative) option in the Event Instruction window allows the user to insert an event into a program that will return an item or reference frame back to a location that was chosen when this event was created. The location will be set in a relative position with respect to the reference frame the item was respecting when the event was created. This option can be chosen in the drop-down menu under the “Action:” setting on the far-left side of the Event Instruction window.
Once this option is chosen, a user can then choose the item that they want to save the current position under the “Select objects to save current position (relative)” part of the Event Instruction window.

When a program is run with a relative position, if the object has been moved, then it will return to the saved location when the event is called. The saved location is the location that the item was at when the event was created by the user. The difference with the relative position, as compared to the absolute, is that if the reference frame has been moved that the object was previously respecting when the event was set up, then the item will return to its saved position with respect to the reference frame that it was defined under.

Once these settings have been chosen, the user can click the “OK” button and the event will be set up.

The location of the object can be updated by modifying the event, which will open the Event Instruction window with the settings that were previously set. The user has to re-select the object in the “Select objects to save current position (relative)” part of the Event Instruction window and then click the “OK” button. The new location will be saved into the event once this has been completed. The modification of the events will be covered by the “Changing the Settings of an Existing Event” section later in the manual.

--Init Program

An init program is a program created by a user when starting to create programs for their station. This is something that is good practice to save time later. The purpose of an init program is to reset the station and return each item, robot arm, and reference frame into the position that was defined by the user when the init was created. An init program is comprised of a series of Set Object position evens, show, or hide evens, and a home target for which the robot will return to when this program is called. The init call is just a suggestion for when a user wants to return a station to its original state and change events and programs that occur during other programs. There is not one way to structure and init program because its structure is completely based on the user’s preference.
--Changing the Settings of an Existing Event

Once an event is created, it can be modified a changed. This is done by right clicking the event that needs to be changed. The user must then click the modify option in the pop-up menu. This brings the user back to the Event Instruction window to edit or redefine what they want the event to do.

-Pause Tool

A pause can be inserted into a program by first highlighting the program in the Station Tree by left clicking on the desired program where the user wants a pause to be. Once it is highlighted, the user can then click the Pause Tool Icon in the Station Toolbar.

A pause will then be added into the program with 500 milliseconds. By clicking on the pause, an Instruction Pause window will pop-up with the option for the user to choose the desired number of milliseconds for the pause.

-Inserting a Python Script

The python scripts are mainly used for controlling mechanisms in the environment such as the conveyor belts. Since it is a mechanism, the python script can control the robot arm if desired; however, it is easier to use the RoboDK tools to manipulate the robot arm. By left clicking and highlighting the desired program that the user wants to place a python script in, the user can choose the Add Python Script Icon in the Station Toolbox.

This places an empty python script under the program that was chosen; however, it does not automatically open a window to edit the script. The user must right click on the python script in the Station Tree, and then choose the Edit Python Code option in the pop-up menu.
This opens a VSCodium window where the user can create or paste python script to control various mechanisms in the station. This manual will not go over coding for such things, but it will show the code that was used in moving the conveyor belts. These will be displayed at the end of the manual with descriptions on how variables are defined and how they work.

-Program Call/Insert Program Instructions

Once smaller programs have been created, they can be called by different programs so the user does not have to remake the similar commands more than once. When a user wants to call a completed program into a different program, they can use the “Program Call or Insert Code Instruction” tool that is found in the Station Toolbar.

To use this tool, the user has to first highlight the program that they are working on, and then they can select this tool in the tool bar which will open the Add Program Call or Custom Code window.

While the main function of this tool is to allow the user to call previously made programs into a different one that the user is constructing, this tool can also be used to call a python script that the user has in the Station Tree. Once the window has been opened, the user must select the Select Program option to pick the program or python code that they want to call. If the user only wants to call one program or python script, then they can select the Okay button and the program call will be inserted.
into the highlighted program. If the user wants to enter more than one program or python call, then they can click next to the program that they selected in the Add Program Call or Custom code, then press the “Enter” key on their keyboard, then they can select the Select Program option again to choose the program call to follow the first one that they entered. This can be repeated for as many times as the user requires. Once they have finished structuring their program and python calls in this window, they can click the Okay button and the program calls will appear in the highlighted program in the order that they structured the program calls in the Add Program Call or Custom Code window.

If the user needs to change the order of program calls or events within a program, they need to first highlight the event that they want to move, then left click and drag that event to the place in the event that the user desires. Once they release the event in the point of the program that the user wants, the event will be moved to that new position which will be shown in the Station Tree.

If the user needs to remove an event or program call from a program, the user needs to highlight that specific event in the program, then press the “Delete” key on their keyboard. This will delete the highlighted event from the program.

-Add Instruction

If the user has a program created with various movements, events, or program calls, the RoboDk software has an option for the user to enter program calls or event between previously existing ones. First, they need to highlight the event in the desired program that happens before the event that they intend to add. If the user requires the event to be the first on in the program, then they can simply highlight the desired program. Then, right click the program to open the pop-up menu and move the cursor over the Add Instruction option.

This opens a second pop-up window that displays all of the event and program addition tools that have been previously discussed in this manual. The user can choose the event that they need to add and follow the instructions for the specific event that have been previously discussed. This adds the specific event behind the highlighted event that the user previously chose. This option is quicker than using the Program Call Instruction tool or any other event tool and dragging the newly created event to the position that the user desires.
-Program Loop

If the user wants to see the same action performed in their RoboDk station, they do not have to call the same program multiple times unless they have a specific number of times that they want the program performed. If the user wants a program on a continuous loop, they can highlight and then right-click the program that they want to loop. This opens a pop-up menu for the chosen program where the user can select the Loop option.

Choosing this option will loop the specific program once the program has been selected to run by the user. Running a program was discussed at the beginning of the Programs section. If a looped program is called within another program, and the larger program is run, then the looped program will start and continue to loop once the larger program has reached the looped program.

-Speed Settings

--Speed Up the Simulation

If a user is running a simulation at normal speed and they want the simulation to run at a faster rate, then they have the option to increase the speed by clicking the Fast Simulation tool in the Station Toolbar.

This tool has a default of 5x speed. If the user wants to increase or decrease the speed, left-clicking the down arrow opens a drop-down window with a slider that displaces the normal speed multiplier value above to the sliding bar.

The user can click on the sliding bar to adjust the value of the normal speed multiplier.

A more precise way of adjusting the speed of the simulation is by hovering the cursor over the Tools option in the Main Menu and selecting the Options option in the drop-down window.
This opens the Options Menu where the user needs to click on the Motion tab at the top of the menu.

This offers a Simulation Time Ratio options where the user can enter speed rates, with respect to normal time, with their keyboard instead of having to use the slider bar.

The user can know that the simulation is running at a faster speed if the Fast Simulation tool turns into a Play tool in the Station Toolbar.

Selecting this tool will return the station to running programs at normal speed.
Next to the Fast Simulation tool is the Pause tool.

Selecting this tool when a program is running stops the program where it was when the user selected the pause tool. The “esc” button can also be used to pause a program when it is running. In order to restart a stopped program, the user must double click the program to restart the simulation.

**Progress Bar**

When a program is double clicked in the Station Tree by the user to run a simulation, a rewind button, pause button, play button, fast-forward button, a next subprogram button, and a progress bar appear in the Status Bar at the bottom of the Station Display.

With the buttons to the right of the status bar, the user can go through a program, reverse the program, jump ahead to the next subprogram, pause, and play the program after it has been started. The statis bar in the middle shows the progress of the simulation. The different colors in the bar highlight the different subprograms that are going to be run during the operating of the main program. Information to the left of the Status bar show the progress of the program that the user can edit. In this part of the Progress Bar, the user can enter a point in the program that they may want to go forward or backward to inside a small window. Note that this number for the progress of the position is not in units of time. Next to the progress adjustment option is the number of seconds that it takes to run the entire program. This is useful to Robot Engineers so that they are able to adjust for the time that it takes the robot to run through the cycle in their manufacturing process, or they can adjust the program or speed of the robot arm so that the time is reduced.

**Gripper Tool Center Point (TCP)**

Before gripper simulation can be covered, the user needs to understand how to think about how the robot interacts with its environment with a gripper attached. This means that the user needs to account for the proper location and orientation of the robot arm so that is can properly grab a specific item.

The Tool Center Point (TCP) of any tool is defining the exact working point of the tool for the robot. This is a problem with a gripping mechanism due to the fingers extending when they close. The TCP for a gripper is defined as the center of the object that the gripper is closing on. RoboDK is limited when defining a changing TCP for a gripper mechanism, and for this purpose, the TCP of the gripper is defined at the point between the fingers when the gripper is fully opened.
As the software simulates picking up different products, the distance from the TCP needs to be accounted for by the user. Since this is a simulated gripper, the distance of the center of the product that the gripper is handling from the TCP defined in RoboDK can be measured with the real gripper by the user and then the user can adjust for this distance in the simulation using the various methods for adjusting the robot’s position and orientation as previously covered.

**Gripper Simulation**

As previously mentioned, the RoboDK is just beginning to offer options for gripper tools that are actual mechanisms in the software. In addition, the software does not yet include a user-friendly way of importing a Solid Works model of a tool and turning it into a mechanism. To account for this, the RoboDK trainings offer a different strategy for simulating gripper operation.

Through their local and online libraries, RoboDK offers many models of grippers that the user might have purchased for their project. If this is the option that suits the user, then reference the Inserting Objects into RoboDK section of this Manual.

If a user wants to import their own gripper into RoboDK from Solid Works, they first need to save their gripper in an open configuration through the method described earlier Local Files Solid Works part of the Inserting Objects into the Station section of this manual. The user would then need to save a more closed version of their gripper in the same fashion. The closed option is to visually simulate the gripper closing around an item. It is recommended, for the look of the simulation, to close the gripper enough to make it look like there is an item between the fingertips. The user can then import both files into their station and place both items on the robot arm flange as explained in the Local Files Solid Work section of this manual. Name each file accordingly so that the user can identify which gripper is closed and which is open. It is best practice to also orientate it to best simulate a physical model. Moving each gripper into position is explained in the Orientation and Positioning section of this manual.

Once the grippers are in place, right click the closed gripper in the Station Tree and de-select the visible option so that it remains a tool, but it is no longer visible. The user can then use a series of program commands and events to show and hide the different gripper configurations along with
attaching and detaching items from them. Instructions for each of these commands are found under the Program section of this manual. This section of the manual will discuss strategy for using these program events to simulate the gripper along with the robot arm. While there are many ways to simulate the gripper, this is the method that was used by the Adaptive Robot Gripper Team.

The user first needs to make a subprogram for picking up an object. The user can name it as they so choose to identify that this program is to pick up an item. The user needs to use the event tool to hide the open gripper, then again to show the closed gripper. Then the user can use the event tool to insert and attach event to the gripper. When the user puts the robot in a configuration such that the gripper is in position to pick up a product, they can call this subprogram in a main program to attach the product to the gripper and make it look like the gripper closed on the product.

The subprogram for the gripper to drop the products is similar to that of the subprograms used to pick up the products. When setting up the program, the user should have the robot holding the product that they want the robot arm to move. It does not have to be any particular object because the attach and detach commands only look for an object that is near the tool chosen by the user. The user should position the robot arm and gripper in the position where it is actually dropping the product. Whether that is on a table that the user imported or a conveyor belt. With dropping, the user should use the insert events tool to hide the closed gripper and then again to show the open gripper. The user then needs to make a detach event where the product respects the reference frame of the object that the item is being placed on.

To simulate time to close and open the gripper, pause events can be used to simulate the robot arm stopping and allowing the gripper to open or close. These events can either be placed before the pick or drop product programs are called, or they can be incorporated in them. This is based on user preference.

Expected Reach of the Robot Arm

After a user has set up their desired robot arm in their station with the tool that they want to use attached to the flange of the robot arm, the RoboDk software allows the user to see the range that the robot arm and tool can reach inside the environment created in the station. To see this option, the user needs to first open the Robot Panel window, which was previously discussed in the Moving the Robot Arm section of this manual. The option to see the range at which the robot arm and tool can reach are found under the Workspace section of the Robot Panel.
Choosing this option will show a sphere with various latitude and longitudinal lines around it. This is the maximum range that the robot arm and tool can reach.

Choosing the Show for Wrist Center option will show the range of the center of the wrist of the robot arm. The Show for Robot Flange option will show the range for the robot flange. The show for current tool will show the range of the tool center point of the tool that has been attached to the robot arm. This option will not be available if there is no tool on the robot arm. The final option is the Do Not Show option, which removes the spheres that shows the reach of the robot arm and tool. As long as the Do Not Show option is not selected, the spheres will continue to be visible in the station. The software also only allows for one of these options to be chosen, so there is no situation where multiple spheres, that show the reach of the various points of the robot arm, can be visible.

The user should note that the sphere that shows the reach of the robot arm is general and does not consider obstacles that are placed within the station. The user should avoid these obstacles when running a program with the use of different events, targets, and orientation settings that have been previously discussed in this manual.
Conveyors

RoboDk offers conveyor mechanisms in their online library. They are still considered mechanisms; however, they are much simpler than the robot arm mechanisms and only have one joint. How the conveyor belt works is that there is a main stationary reference frame for the conveyor belt that defines the conveyor’s stationary position within the station. The joint controls for this mechanism are used to move a reference frame along the belt to simulate objects moving on the top of the conveyor belt like they would on a manufacturing line.

To move these, RoboDk has python scripts that control the number of millimeters the movable reference frame travels along the belt and in what direction the reference frame moved. This given python code will be displayed in the Python Scripts used for the Adaptive Robot Gripper part of this manual along with an explanation of how to use it. A third script was used to reset the conveyor into its original position. This follows the recommendation about init programs that was discussed under the Programs section of this manual. All of the codes worked the same for any conveyor belt model taken from the RoboDK online library. The only changes that needed to be made to the python scripts are the mechanisms that they control. This will also be discussed in the Python Scripts section of the manual where the code is displayed.

The conveyor was the only mechanism in the Adaptable Robot Gripper simulation that used a python script call in the Station Tree. The python codes were called by different programs to move the conveyors in a certain direction at a specific time in the simulation. It was assumed that the conveyor belts in the simulation would stop intermittently. This is due to the progress that was made in the Adaptive Robot Gripper project along with the limits of the RoboDK software. There was no example of a simulated a conveyor belt that did not stop and there was not enough time in the project to experiment with the simulation software.

Note that if this software were to be used to control a robot in a manufacturing line, the software does not have the ability to control a conveyor also, this is just for the simulation and timing of the robot arm.

Python Scripts Used in the Adaptive Robot Gripper Project

Descriptions for creating python scripts in the Station Tree can be found in the Inserting a Python Script part of the Programs section of this manual. It is best to first give the python code a name that symbolizes its function. Naming items in the Station Tree is talked about in the Renaming section of this manual. Once a script is created and renamed in the Station Tree, a user can insert and edit code by highlighting and right clicking the python code that the user wants to edit in the Station Tree. This will open a pop-up menu where the user must click the Edit Python Script option.
This opens a VSCodium window where the user can insert code if it was a blank python script or change the code if they had already inserted code into it.

The following python codes can be found in the trainings from RoboDK. Each will have a title, show their code, and have a description of what they do and things that can be changed by the user as they desire for their specific needs.

Title: InitCovPosition

Code:

```python
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *     # RoboDK API
from robodk import *       # Robot toolbox
RDK = Robolink()

MECHANISM_NAME = 'Conveyor_In'
mechanism = RDK.Item(MECHANISM_NAME,itemtype=ITEM_TYPE_ROBOT)

if mechanism.Valid():
    mechanism.setJoints([0])
```
MECHANISM_NAME = 'Conveyor_Out'
mechanism = RDK.Item(MECHANISM_NAME,itemtype=ITEM_TYPE_ROBOT)

if mechanism.Valid():
    mechanism.setJoints([0])

Description:

This code was made to reset the position of the conveyor belts. As seen in the code, there are two mechanisms, one titled “Conveyor_In” and the other title “Conveyor_Out”. The first is the name of the conveyor that brings the products in front of the robot arm, and the second mechanism is the conveyor that the robot arm drops the products on. For each mechanism, the code defines the type of mechanism and sets their joint to the starting position, or in terms of the conveyor belts in this software, brings the movable reference frames to the starting location. See the conveyor section of this manual for a more in-depth description.

Title: NextPart_In

Code:

# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
RDK = Robolink()

#------ CONSTANT ------
MECHANISM_NAME = 'Conveyor_In'
PART_TRAVEL_MM = 1750

mechanism = RDK.Item(MECHANISM_NAME,itemtype=ITEM_TYPE_ROBOT)

if mechanism.Valid():
    mechanism.MoveJ(mechanism.Joints() + PART_TRAVEL_MM)

Description:

This code was specifically designed to only move the conveyor bringing products to the robot arm. This is shown by the defined mechanism name “Conveyor_In”. The line of code below the defining
of the mechanism name is the travel distance of the conveyor. For the code shown, each time the python code is called in the station, the movable reference frame will move 1750 millimeters. This can be changed as the user sees fit.

Title: NextPart_Out

Code:

```python
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Note: It is not required to keep a copy of this file, your python script is saved with the station
from robolink import *  # RoboDK API
from robodk import *  # Robot toolbox
RDK = Robolink()

#------ CONSTANT ------
MECHANISM_NAME = 'Conveyor_Out'
PART_TRAVEL_MM = 2000

mechanism = RDK.Item(MECHANISM_NAME, itemtype=ITEM_TYPE_ROBOT)

if mechanism.Valid():
    mechanism.MoveJ(mechanism.Joints() + PART_TRAVEL_MM)
```

Description:

This code was specifically designed to only move the conveyor that the robot arm drops products off onto. This is shown by the defined mechanism name “Conveyor_Out”. The line of code below the defining of the mechanism name is the travel distance of the conveyor. For the code shown, each time the python code is called in the station, the movable reference frame will move 2000 millimeters. This can be changed as the user sees fit.

Opening Robot Codes Based on Station Programs

RoboDk has the option of producing code, in the language for the robot chosen in the RoboDK station, for the programs created by the user in their station. These codes can be created and exported to the Raspberry Pi controller so that the entire Adaptive Robot Gripper Software can communicate with the Robot Arm. This section will show the RoboDK user how to pull up the code for the programs created in their RoboDK station.
Once the programs have been structured as the user desires, the user can go to the Main Menu of their station and hover their cursor over the program option. This opens a drop-down menu where they can then choose the Generate Program option.

A menu then pops up so that the user can choose the specific program that they want generated for the language of the chosen robot.

Once a program was chosen and the Okay button is clicked, the code for the robot is displayed in a VSCodium window. This is to show the potential of this software. It has the ability to control the robot online and offline; however, interfacing and connecting to the robot is outside of the scope of this project since there was not a chance to actually work with a robot arm.
Adaptive Robot Gripper
Technical Data Package
Version: 3

Team: Riyadh Alswayel, Max Critchfield, Lucas Hawley, Joe Liang, Connor Nagore
Sponsor: Unilever
Technical Data Package

Table of contents

1. Project Description ..............................................................................................................99
2. Functional Requirements ..................................................................................................99
3. System Verification Plan ....................................................................................................100
   a. SRVM ..........................................................................................................................100
   b. Verification Matrix and Results ................................................................................101
   c. Verification Procedure ..............................................................................................102
4. Design Documentation .....................................................................................................124
   a. Hardware Drawing Package ....................................................................................124
   b. IDL .............................................................................................................................132
   c. System Requirements Documentation ......................................................................130
      i. System block diagram .........................................................................................130
      ii. System Architecture ..........................................................................................131
   d. Software Documentation ..........................................................................................140
5. Appendix ..........................................................................................................................32
Description

The Adaptive Robot Gripper (ARG) project has the purpose of designing and producing a prototype of a universal gripper for Unilever production lines. This device is meant to have extended capability beyond their current industrial robot grippers and will provide the ability to pick up products of various shapes, sizes, and weights as well as have capability to pick up products with differing package rigidity without damage to the packaging.

The ARG will be comprised of a camera system, a gripper, and emulation software. The camera system will identify the location of the product while the size and shape of the recognized item will be retrieved from stored data in the Raspberry Pi. This information will be sent to the arm and gripper so that the product will be handled carefully. The included emulation software will simulate the industrial robot arm and gripper in the manufacturing environment. The virtual simulation of the arm will show how the robot arm would respond to the location of products on a conveyer belt and show how it would orientate itself to pick up and drop each Unilever Product.

Functional requirements

1. The ARG shall be used in an industrial/assembly line environment.
2. The ARG shall grab and release objects when appropriate.
3. The ARG shall be able to pick up items of different: sizes, shapes, densities, and weights.
4. The ARG shall connect/interface with existing robot arms.

The ARG shall function without a human operator.
### Requirements:

<table>
<thead>
<tr>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2.1 Camera System</strong></td>
</tr>
<tr>
<td>2.1.1 Object recognition: Camera system shall detect and recognize the type of object to be gripped (95% accuracy)</td>
</tr>
<tr>
<td>2.1.2 Stationary: The camera system shall be stationary next to the conveyer system &amp; the ARG shall have a default location</td>
</tr>
<tr>
<td><strong>2.2 Sensors and software</strong></td>
</tr>
<tr>
<td>2.2.1 Data lookup: Pi shall retrieve all relevant data for each object</td>
</tr>
<tr>
<td>2.2.2 Distance Detection: The ultrasonic sensor shall report object distances. (+/- 5mm)</td>
</tr>
<tr>
<td>2.2.3 Force Sensor: The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips</td>
</tr>
<tr>
<td>2.2.4 Emulation: the ARG shall utilize emulation software in place of a robot arm for demonstration purposes</td>
</tr>
<tr>
<td><strong>2.3 Performance and cost</strong></td>
</tr>
<tr>
<td>2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000</td>
</tr>
<tr>
<td>2.3.2 Performance: The ARG shall have a success rate greater than 99%.</td>
</tr>
<tr>
<td>2.3.3 Pick rate: The ARG shall have a pick rate of at least 30 picks per minute.</td>
</tr>
<tr>
<td><strong>2.4 Mechanical and Design</strong></td>
</tr>
<tr>
<td>2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products</td>
</tr>
<tr>
<td>2.4.2 Handle: the ARG will have a handle for demonstration</td>
</tr>
<tr>
<td>2.4.3 Safety: The ARG shall have a shut down mechanism for emergency situations</td>
</tr>
<tr>
<td>Requirements</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>2.1 Camera System</td>
</tr>
<tr>
<td>2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolov5 neural network model (95% accuracy)</td>
</tr>
<tr>
<td>2.1.2 Stationary: The camera system shall be stationary next to the conveyor system &amp; the ARG shall have a default location</td>
</tr>
<tr>
<td>2.2 Sensors and software</td>
</tr>
<tr>
<td>2.2.1 Data lookup: PI shall retrieve all relevant data for each object.</td>
</tr>
<tr>
<td>2.2.2 Distance Detection: The ultrasonic sensor shall report object distances: (+/- 3mm)</td>
</tr>
<tr>
<td>2.2.3 Force Sensor: The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips</td>
</tr>
<tr>
<td>2.2.4 Simulation: the ARG shall utilize simulation software in place of a robot arm for demonstration purposes</td>
</tr>
<tr>
<td>2.3 Performance and cost</td>
</tr>
<tr>
<td>2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000.</td>
</tr>
<tr>
<td>2.3.2 Performance: The ARG shall have a success rate of 95%.</td>
</tr>
<tr>
<td>2.3.3 Pick rate: The ARG shall have a pick rate of at least 30 picks per minute. (0.6 seconds/grip)</td>
</tr>
<tr>
<td>2.4 Mechanical and Design</td>
</tr>
<tr>
<td>2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products</td>
</tr>
<tr>
<td>2.4.2 Handle: the ARG will have a handle for demonstration</td>
</tr>
<tr>
<td>2.4.3 Safety: The ARG shall have a shut-down mechanism for emergency situations</td>
</tr>
</tbody>
</table>
Object Identification Test:

Requirement and Justification:

2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolov5 neural network model (95% accuracy)

To reach the 95% threshold outlined above, the ability of our camera system to correctly identify each of our test objects must be tested.

Procedure:

5. Set up a controlled imaging environment. Mount the camera in a stationary position such that the viewing background is uniform and with appropriate lighting.
6. Place an object in the camera’s field of view.
7. Read the software output to confirm the object was correctly identified with at least 70% confidence.
8. Complete steps 3 and 4 at least 3 times for each of the 5 objects. Record the results.

The justification for the 70% passing metric comes from the software training process we used to identify our objects. We used over 1000 images to train the software and any object that was identified with a confidence of 70% or more was always a correct identification.
# 2.1.1 Acceptance Test Data Sheet

Referenced ATP Paragraph Number: 2.1.1

Analysis Referenced: N/A

Name of Test: Object ID

Unit Under Test (UUT):
Name: Visualization System
Part Number: 101000, 102101

<table>
<thead>
<tr>
<th>Results (Pass/Fail)</th>
<th>Date of Test: 2/15/22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

Recording of Test Measurement:

<table>
<thead>
<tr>
<th>Object</th>
<th>Requirement (with tolerances):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Must be recognized with &gt;70% confidence</td>
</tr>
</tbody>
</table>

|          | Readout 1: 75, 86, 82 – Pass |
|          | Readout 2: 79, 85, 87 – Pass |
|          | Readout 3: 76, 83, 86 – Pass |
|          | Readout 4: 91, 88, 92 – Pass |
|          | Readout 5: 89, 87, 90 – Pass |

Smallest Margin: 5%
Average Margin: 15%
PASS

Computations and Notes:

* 70% metric was decided through testing. For our 1000+ training image any that had 70%+ confidence correctly identified the object.
**Depth of Field Demonstration:**

**Requirement and Justification:**

2.1.1 Object recognition: Camera system shall detect and recognize the type of object (bodywash, deodorant, spray, soap, tea) to be gripped using the trained yolov5 neural network model (95% accuracy)

In order to operate in an industrial environment, the camera must be able to identify an object across the entire width of the conveyor transporting it. A 1-meter depth of field will ensure this capability. Depth of field is the distance between the nearest and the farthest objects that are in acceptably sharp focus in an image. In order to pass this demonstration, the camera system must correctly identify the object with the same 70% confidence from the object identification test across a 1m length without changing the camera's focus.

**Method and Data:**

We tested each object at 3 points: one at the closest point where the objects are in the field of view (z=0m), another .5 meters away from the first point (z=.5m), and finally a point 1 meter away from the first (z=1m). The results are below:

<table>
<thead>
<tr>
<th>z</th>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0m</td>
<td>Tea – 0.800</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.847</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.752</td>
</tr>
<tr>
<td></td>
<td>Soap 0.786</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.930</td>
</tr>
<tr>
<td>.5m</td>
<td>Tea – 0.941</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.917</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.906</td>
</tr>
<tr>
<td></td>
<td>Soap 0.888</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.845</td>
</tr>
<tr>
<td>1m</td>
<td>Tea – 0.928</td>
</tr>
<tr>
<td></td>
<td>Bodywash – 0.915</td>
</tr>
<tr>
<td></td>
<td>Spray – 0.829</td>
</tr>
<tr>
<td></td>
<td>Soap 0.903</td>
</tr>
<tr>
<td></td>
<td>Deodorant – 0.837</td>
</tr>
</tbody>
</table>

All values are greater than 70% or .7 so this demonstration was successful.
Stationary Inspection:

Requirement:

2.1.2 Stationary: The camera system shall be stationary next to the conveyer system & the ARG shall have a default location

Justification and Result:

In order to get consistent results and data, the camera system and ultrasonic sensor must be mounted statically. The image below shows the mount we made to achieve this goal.

Despite being made of LEGO parts the above mount is a cheap and sturdy housing for these two components of our system and satisfies requirement 2.1.2.
Data Lookup Demonstration:

Requirement:

2.2.1 Data lookup: Pi shall retrieve all relevant data for each object

Justification and Result:

Following a successful object identification, the software must be able to access the appropriate preprogrammed data from its code. The image below shows the relevant section of code which we have tested thoroughly.

Following the object identification, the respective set of commands will be followed.
Distance Detection Test:

Requirement and Justification:

2.2.2 Distance Detection: The ultrasonic sensor shall report object distances. (+/- 5mm)

In order to give location data to the robot arm responsible for positioning the gripper, a distance sensor will determine objects position on the conveyor belt. Since the grip heights have been preprogrammed and the conveyor will stop in the same position each time, this sensor only needs to operate on one axis, the lateral position of the object on the belt. The following test was conducted to ensure the accuracy of this detector.

Procedure:

7. Set up the ultrasonic sensor in a stationary position with at least 1.5 meters of space in front of it.
8. Place an object some distance in front of the ultrasonic sensor.
9. Measure the distance from the sensor to the object manually. (Metric measurement is preferred to avoid unit conversions)
10. Read the software output with sensed measurement.
11. Compare outputs from steps 4 and 5. If they are within 5mm of one another, this trial is a pass.
12. Complete steps 2-5 at least 5 times. Record the results.
### 2.2.2 Acceptance Test Data Sheet

**Reference ATP Paragraph Number:** 2.2.2  

**Analysis Referenced:** N/A  

**Name of Test:** Distance Test  

**Unit Under Test (UUT):**  
- **Name:** Ultrasonic Sensor  
- **Part Number:** 105200  

**Results (Pass/Fail):** Pass  

**Date of Test:** 2/17/22  

**Recording of Test Measurement:**  

<table>
<thead>
<tr>
<th>Measured Distance (mm)</th>
<th>Requirement (with tolerances):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 304.8</td>
<td>Must be with +/- 5 mm</td>
</tr>
<tr>
<td>2. 254</td>
<td>Sensed Distance (mm)</td>
</tr>
<tr>
<td>3. 203.2</td>
<td>1. 302.8 (-2 mm): Pass</td>
</tr>
<tr>
<td>4. 101.6</td>
<td>2. 254.4 (-2.6 mm): Pass</td>
</tr>
<tr>
<td>5. 50.8</td>
<td>3. 202.8 (-.5 mm): Pass</td>
</tr>
<tr>
<td></td>
<td>4. 101.6 (0 mm): Pass</td>
</tr>
<tr>
<td></td>
<td>5. 50.7 (-.1mm): Pass</td>
</tr>
</tbody>
</table>

**Smallest Margin:** 2.4mm  

**Average Margin:** 2.96mm  

**PASS**  

**Computation and Notes:**  

1 Inch = 2.54 cm, 1 cm = 10 mm
Force Sensor Calibration Test:

Requirement and Justification:

2.2.3 Force Sensor: The ARG shall be able to apply the appropriate force to the object by having the ability to accurately measure the force it is currently applying. (±1) N

The initial purpose for the FSR was to ensure the object was not gripped too tightly or too loosely to avoid crushing or dropping the object. From the test sheet below it can be seen that this part failed to meet the intended specification. This being the case, we repurposed the FSR to only monitor whether or not the object has been dropped so the system could be shut down in case of failure. Despite the FSR no longer setting the upper limit of grip force, crushing the objects remains a non-issue since the motors used in the adaptive robot gripper are not strong enough to damage any of the objects under test. Even with this new purpose, the force sensitive resistor used for the gripper is inadequate. For future iterations of the gripper, a new one should be selected.

Procedure:

9. Set up the force sensitive resistor (FSR) stationarily on a flat surface.
10. Use a scale to measure the mass of your test object. (54 grams for the data below)
11. Convert measured mass to gravitational force using equation 1 on the test sheet below.
12. Place test object on the sensor.
13. Read the software output with sensed measurement in bits.
14. Convert bit value to force value using equation 2 on the test sheet below.
15. Compare outputs from steps 3 and 6. If they are within 1N of one another, this trial is a pass.
16. Complete steps 2-5 at least 5 times. Record the results.
**2.2.3 Acceptance Test Data Sheet**

Referenced ATP Paragraph Number: 2.2.2

Analysis Referenced: N/A

Name of Test: Force Calibration Test

Unit Under Test (UUT):
Name: Force Sensor  
Part Number: 105100

Results (Pass/Fail): **Fail**  
Date of Test: **4/12/22**

### Measured force (N)

<table>
<thead>
<tr>
<th>Weight (g)</th>
<th>Sensed Force (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>216</td>
<td>2.12</td>
</tr>
<tr>
<td>270</td>
<td>2.65</td>
</tr>
<tr>
<td>324</td>
<td>3.18</td>
</tr>
<tr>
<td>378</td>
<td>3.71</td>
</tr>
<tr>
<td>432</td>
<td>4.24</td>
</tr>
</tbody>
</table>

### Requirement (with tolerances):
Must be within 1 N

<table>
<thead>
<tr>
<th>Sensed Force (N)</th>
<th>Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.31</td>
<td>+2.19N</td>
</tr>
<tr>
<td>4.89</td>
<td>+2.23N</td>
</tr>
<tr>
<td>2.11</td>
<td>-1.07N</td>
</tr>
<tr>
<td>2.59</td>
<td>-1.12N</td>
</tr>
<tr>
<td>2.30</td>
<td>-1.94N</td>
</tr>
</tbody>
</table>

**Average Error:** 1.71N  
**Average % Error:** 71%

**FAIL**

### Computations and Notes:

1. 
2. 
3. $N = m \cdot 9.81 \text{ms}^{-1}$
4. $N=(10\text{kg} \cdot \text{nbits} \cdot 9.81\text{ms}^{-1})/1024$
Engineering Change Request

Date: 19 April 2022

Team Number 22053  Project Name: Adaptive Robot Gripper

Title of Change: Force Sensor Requirement 2.2.3

Documents Affected (check all that apply):

<table>
<thead>
<tr>
<th>Doc Type</th>
<th>Document Number</th>
<th>Document Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ SOW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ SRD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ SDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ TP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ EVMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☐ Drawing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>☒ Other</td>
<td></td>
<td>FAR and Final Report</td>
</tr>
</tbody>
</table>

Description of Change: Requirement 2.2.3 states, “Force Sensor: The ARG shall be able to apply the appropriate force to the object by having the ability to measure the force it is currently applying.” While we will still use the force sensor, its purpose will be to check whether or not the product has been grabbed. This will remove the force sensor feedback loop for the gripper and the information of whether the gripper has picked the item will only be sent to the GUI. Due to this change, we would also like to change this requirement from being a test to a display.

Need for Change: The force sensors that we have been working with have not been able to read the force being applied to the products consistently and accurately.

Impact: This will remove the force sensor feedback from our system and replace it with a check on whether or not the gripper has picked up the item. Since this is intended to be on an assembly line and items are on a conveyor belt, the gripper and robot arm will not be affected by this signal that tells the software if the item has been grabbed.

Substantiation: This is just to remove the force sensor feedback loop. Our system is still able to perform with out it and not damage the items. The only change is that the gripper will lose some precision with the removal of this sensor feedback. The system will rely upon the pre-coded data to properly grab the products that it recognizes.
**Force Sensor Drop Detection Demonstration:**

**Requirement and Justification:**

2.2.3 Force Sensor: The ARG shall be able to detect whether an object has been dropped by monitoring the force applied to the fingertips

The initial purpose for the FSR was to ensure the object was not gripped too tightly or too loosely to avoid crushing or dropping the object. From the test sheet above it can be seen that this part failed to meet the intended specification. This being the case, we repurposed the FSR to only monitor whether or not the object has been dropped so the system could be shut down in case of failure. Despite the FSR no longer setting the upper limit of grip force, crushing the objects remains a non-issue since the motors used in the adaptive robot gripper are not strong enough to damage any of the objects under test. Even with this new purpose, the force sensitive resistor used for the gripper is inadequate due to inconsistent readings. For future iterations of the gripper, a new one should be selected.

**Procedure:**

For this demonstration, the gripper holds an object which is then manually removed from the fingers while it is still meant to be held. The force values output by the sensor are monitored during this process. A consistent change from a high to low threshold must be observed so these thresholds can be coded into the project as ‘held’ and ‘dropped’ states.
<table>
<thead>
<tr>
<th>Recorded Test Measurement:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured force while held (bits)</td>
</tr>
<tr>
<td>1. 122</td>
</tr>
<tr>
<td>2. 100</td>
</tr>
<tr>
<td>3. 131</td>
</tr>
<tr>
<td>4. 85</td>
</tr>
<tr>
<td>5. 90</td>
</tr>
</tbody>
</table>

Requirement (with tolerances):

Must show demonstrable drop in force output

Computation and Notes:

Despite a general downturn in detected force, there is too much overlap between the outputs between held and dropped for a reliable distinction to be made.
Emulation Demonstration:

Requirement and Justification:

2.2.4 Emulation: the ARG shall utilize emulation software in place of a robot arm for demonstration purposes

Since a robot arm could not be sourced for this project, we will be manually positioning the gripper with a handle we’ve affixed to it. To demonstrate how the gripper will work in an industrial environment, we will show an emulation of our gripper on a robot arm operating on a conveyor belt. A screenshot is attached below.
Cost Analysis:

Requirement and Justification:

2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000

All senior design teams are given the same $4000 budget they must stay within. This makes budgeting and planning extremely important to ensure the project can be completed. A full breakdown of the budget for the adaptive robot gripper is shown below. There is also a second figure showing the difference between the total budget which includes extra parts, prototypes, and other development costs as well as the system cost.

<table>
<thead>
<tr>
<th>Item</th>
<th>Supplier</th>
<th>Qty</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 8GB RAM</td>
<td>Labists</td>
<td>2</td>
<td>302.98</td>
</tr>
<tr>
<td>Camera Flex Cable</td>
<td>Amazon</td>
<td>1</td>
<td>8.79</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Amazon</td>
<td>1</td>
<td>12.99</td>
</tr>
<tr>
<td>Buttons 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>6.39</td>
</tr>
<tr>
<td>Stepper Motor Driver 5pcs</td>
<td>Amazon</td>
<td>1</td>
<td>21.49</td>
</tr>
<tr>
<td>Stepper Motor Wires</td>
<td>Amazon</td>
<td>1</td>
<td>8.99</td>
</tr>
<tr>
<td>12V Power Supply</td>
<td>Amazon</td>
<td>2</td>
<td>29.80</td>
</tr>
<tr>
<td>Pressure Sensors 2pcs</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Analog to Digital Converter</td>
<td>Amazon</td>
<td>3</td>
<td>13.65</td>
</tr>
<tr>
<td>Ultrasonic Sensor</td>
<td>Amazon</td>
<td>1</td>
<td>13.99</td>
</tr>
<tr>
<td>RoboDK License</td>
<td>RoboDK</td>
<td>1</td>
<td>145.00</td>
</tr>
<tr>
<td>Electrical Hardware Hit</td>
<td>Amazon</td>
<td>1</td>
<td>18.49</td>
</tr>
<tr>
<td>Jumper Wires</td>
<td>Amazon</td>
<td>1</td>
<td>25.95</td>
</tr>
<tr>
<td>Metal Gripper</td>
<td>Xometry</td>
<td>1</td>
<td>1361.29</td>
</tr>
<tr>
<td>Plastic Gripper</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>88.11</td>
</tr>
<tr>
<td>Handle</td>
<td>UA 3D Printing</td>
<td>1</td>
<td>9.97</td>
</tr>
<tr>
<td>Gripping Material</td>
<td>Amazon</td>
<td>2</td>
<td>25.98</td>
</tr>
<tr>
<td>Nema Stepper Motors</td>
<td>Amazon</td>
<td>5</td>
<td>84.15</td>
</tr>
<tr>
<td>M3 Screws</td>
<td>Amazon</td>
<td>1</td>
<td>11.49</td>
</tr>
<tr>
<td>Monitor</td>
<td>Best Buy</td>
<td>1</td>
<td>299.99</td>
</tr>
<tr>
<td>Conveyor Belt</td>
<td>Vevor</td>
<td>1</td>
<td>248.99</td>
</tr>
<tr>
<td>Assorted Zip Ties Pack</td>
<td>Amazon</td>
<td>1</td>
<td>15.95</td>
</tr>
<tr>
<td>Poster</td>
<td>Print Shop</td>
<td>1</td>
<td>110.00</td>
</tr>
<tr>
<td>Shirts</td>
<td>Engineering College</td>
<td>5</td>
<td>150.00</td>
</tr>
<tr>
<td>Raspberry Pi HQ Camera</td>
<td>Vilros</td>
<td>1</td>
<td>50.00</td>
</tr>
<tr>
<td>6mm Camera Lens</td>
<td>Amazon</td>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>Total (Excluding Tax and Shipping)</td>
<td></td>
<td></td>
<td>3114.43</td>
</tr>
<tr>
<td>Total (Including Tax and Shipping)</td>
<td></td>
<td></td>
<td>3358.96</td>
</tr>
</tbody>
</table>
### Project Costs

<table>
<thead>
<tr>
<th>Category</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical/Software</td>
<td>633.51</td>
</tr>
<tr>
<td>Mechanical</td>
<td>1580.99</td>
</tr>
<tr>
<td>Display</td>
<td>824.93</td>
</tr>
<tr>
<td>Optical</td>
<td>75.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3114.43</strong></td>
</tr>
<tr>
<td><strong>Total (Tax, Shipping)</strong></td>
<td><strong>3358.96</strong></td>
</tr>
</tbody>
</table>

---

**Performance Test:**

**Requirement and Justification:**

2.3.2 Performance: The ARG shall have a success rate of 95%.

For a product that will be used in a production line handling up to 30 objects per minute, it is important to have a success rate that is as close to perfect as possible. Even with a 99.9% success rate the gripper would fail more than 14 times in an 8-hour period. This being the case, we were initially tasked with getting as close to 100% as possible, but with less than a year to develop the gripper and the fact that 100% accuracy is not something we could measure in a way that is statistically significant, Unilever agreed to reduce it to 95% so that we could properly measure it. To test this metric, we have to test the system in its entirety. Therefore, a success can only be recorded if all of the following steps are successful:

7. The distance detector must trigger when an object is present to start the process.
8. The camera must image the object for the software to identify.
9. The object must be identified correctly by the software.
10. The object must be moved down the conveyor to the gripping plane.
11. The gripper must pick up the object successfully.
12. The gripper must release the object in the placement zone.

**Procedure:**

8. Start the code.
9. Place an object in the detection plane.
10. Wait for camera to identify the object.
11. Activate the conveyor belt.
12. Position the gripper in the correct orientation for the object.
13. Move object to placement zone.
14. Repeat above steps 5 times for each object.

The results for this test are displayed on the datasheet below.
# 2.3.2 Acceptance Test Data Sheet

<table>
<thead>
<tr>
<th>Referenced ATP Paragraph Number: N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis Referenced: N/A</td>
</tr>
<tr>
<td>Name of Test: Adaptability Demonstration</td>
</tr>
<tr>
<td>Unit Under Test (UUT):</td>
</tr>
<tr>
<td>Name: Performance Test</td>
</tr>
<tr>
<td>Part Number: Whole System</td>
</tr>
<tr>
<td>Results (Pass/Fail): Pass</td>
</tr>
<tr>
<td>Recording of Demonstration Measurement:</td>
</tr>
<tr>
<td>Object 1: BODYWASH</td>
</tr>
<tr>
<td>Object 2: DEODORANT</td>
</tr>
<tr>
<td>Object 3: SOAP</td>
</tr>
<tr>
<td>Object 4: SPRAY</td>
</tr>
<tr>
<td>Object 5: TEA</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Pass rate: 96%

Computations and Notes:

system pass is defined on the previous page.

Drops from user error were omitted.
**Pick Rate Demonstration:**

Requirement and Justification:

2.3.3 Pick rate: The ARG shall have a pick rate of at least 30 picks per minute. (0.6 seconds/grip)

With the high volume of products moving down the line at Unilever manufacturing plants, time is money. The more objects that can be gripped in a given time period, the more products make it out the door. This being the case it was important that the gripper could facilitate 30 picks per minute. A ‘pick’ is defined as the process of grabbing the object, moving it to its new location, setting it down, and returning to the initial position to grab the next object. In this process the majority of the time, 1.4 seconds in the case of Unilever robot arms, is spent moving back and forth. This leaves 0.6 seconds for the gripper to close around the object and open again to release it. Since we were without an assembly line to test this full scale in its final environment, we timed the opening and closing process for each of the objects and recorded these values. So long as it was able to open and close securely around an object in less than 0.6 seconds, it was considered a successful demonstration. The demonstration sheet with these results is displayed below.

<table>
<thead>
<tr>
<th>2.2.3 Acceptance Test Data Sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced ATP Paragraph Number: N/A</td>
</tr>
<tr>
<td>Analysis Referenced: N/A</td>
</tr>
<tr>
<td>Name of Test: Pick Rate Test</td>
</tr>
<tr>
<td>Unit Under Test (UUT):</td>
</tr>
<tr>
<td>Name: Mechanical Gripper</td>
</tr>
<tr>
<td>Part Number: 104100,104302</td>
</tr>
<tr>
<td>Results (Pass/Fail): <strong>Pass</strong></td>
</tr>
<tr>
<td>Recording of Demonstration Measurement:</td>
</tr>
<tr>
<td>Object 1: SOAP</td>
</tr>
<tr>
<td>Object 2: DEODORANT</td>
</tr>
<tr>
<td>Object 3: BODYWASH</td>
</tr>
<tr>
<td>Object 4: SPRAY</td>
</tr>
<tr>
<td>Object 5: TEA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Adaptability Demonstration:

Requirement and Justification:

2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products

The gripper must be able to grip five unique Unilever products that vary in shape and weight: Axe deodorant, Dove bodywash, Lipton tea, Dove bar soap, and Axe body spray. While the overall performance of the system is tested in the performance test, this preceding demonstration was simply meant to show that the gripper was capable of picking up each object. The demonstration sheet and images of each object being held by the gripper are displayed below.

<table>
<thead>
<tr>
<th>2.4.1 Acceptance Test Data Sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced ATP Paragraph Number: N/A</td>
</tr>
<tr>
<td>Analysis Referenced: N/A</td>
</tr>
<tr>
<td>Name of Test: Adaptability Demonstration</td>
</tr>
<tr>
<td>Unit Under Test (UUT):</td>
</tr>
<tr>
<td>Name: Adaptability Demonstration</td>
</tr>
<tr>
<td>Part Number: 104100, 104300</td>
</tr>
<tr>
<td>Results (Pass/Fail): Pass</td>
</tr>
<tr>
<td>Recording of Demonstration Measurement:</td>
</tr>
<tr>
<td>Object 1: BODYWASH</td>
</tr>
<tr>
<td>Object 2: DEODORANT</td>
</tr>
<tr>
<td>Object 3: SOAP</td>
</tr>
<tr>
<td>Object 4: SPRAY</td>
</tr>
<tr>
<td>Object 5: TEA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Computations and Notes:</td>
</tr>
<tr>
<td>Gripper is capable of grabbing all objects.</td>
</tr>
</tbody>
</table>
Dove Bodywash

Dove Bar Soap

Axe Body Spray

Lipton Tea

Axe Deodorant
Handle Demonstration:

Requirement and Justification:

2.4.2 Handle: the ARG will have a handle for demonstration

Since the team does not have a robot arm a handle is needed to facilitate demonstration of the system. Using this handle, we will be able manually position the gripper as accurately as we can. The SolidWorks file showing the handle is displayed below.
Safety Demonstration:

Requirement and Justification:

2.4.3 Safety: The ARG shall have a shut-down mechanism for emergency situations

In order to make the system as safe as possible, a shut-off button has been implemented. Upon the press of the button, the system will stop all processes immediately. The wire diagram for this button is shown below.

While very simple, this setup is an effective input method to initiate the shut-off.
Design Documentation IDL:

Figure.1 Palm Part

Figure.2 Link Part
Figure 5. Grip Part

Figure 6. Drive Gear Part
Figure 8. The assembly of the adaptive robot gripper
Figure 9. The assembly of the adaptive robot gripper (exploded view)
System Requirements Documentation:

System Block Diagram:

Team Responsibilities:
- Raspberry Pi 4
  - Object detection and ultrasonic location software
  - Motor driver and force sensor software

Eligibility:
- display GUI of object recognition and sensor data
- Display robot simulation

Outlet/Power Supply:
- 120 V (~80,000 mAh battery)
- USB to USB, O2, Wi-Fi

Flexible printed circuit (FPC):
- 5.1 V (3.0 USB)

Raspberry Pi 4 Block Diagram:
- Object detection and ultrasonic location software
- Motor driver and force sensor software

Motor Driver:
- Controls opening and closing of mechanical gripper

Mechanical Gripper System:
- 2x Gripper Fingers
- 2x Grip Plates
- 2x Gear Links
- 1x Gripper Palm
- 2x Links

Grip Plate Anti-Slip Material:
- Pressure sensitive

Robot Tech F134812
- Force Sensors
- 9kg/20 lbs sensitivity
- Force feedback

Object:
- 5 different products
- >2 different products for testing failures

Object placed by Mechanical Gripper

Object begins in visualization plane on conveyor system

Object moves to conveyor system
- Picking Zone
- Object stops at pickup location
System Architecture:

Adaptive Robot Gripper Architecture

1.0 Adaptive Robot Gripper (ARG)

1.1 Camera System
- 1.1.1 Camera
- 1.1.2 Camera Assembly

1.2 Computer Unit
- 1.2.1 Raspberry Pi
- 1.2.2 GUI
- 1.2.3 Object Detection API
- 1.2.4 Gripper Driver Software
- 1.2.5-1.2.6 Sensor Software
- 1.2.7 Camera API

1.3 Frame Set
- 1.3.1 Base Plate
- 1.3.2 Interfaces

1.4 Gripper
- 1.4.1 End Point Gripper
- 1.4.2 Gripper Shell
- 1.4.3 Motors
- 1.4.4 Wrist Assembly

1.5 Sensors
- 1.5.1 Force Sensor
- 1.5.1 Ultrasonic Sensor
## Parts List:

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10100</td>
<td>Camera System</td>
</tr>
<tr>
<td>101100</td>
<td>Camera</td>
</tr>
<tr>
<td>101101</td>
<td>Raspberry Pi HQ Camera</td>
</tr>
<tr>
<td>101200</td>
<td>Camera Assembly</td>
</tr>
<tr>
<td>101201</td>
<td>6mm Wider Angle Lens</td>
</tr>
<tr>
<td>102000</td>
<td>Computer Unit</td>
</tr>
<tr>
<td>102100</td>
<td>Raspberry Pi</td>
</tr>
<tr>
<td>102101</td>
<td>Raspberry Pi 4 8GB</td>
</tr>
<tr>
<td>102200</td>
<td>GUI</td>
</tr>
<tr>
<td>102300</td>
<td>Object Detection Software Package</td>
</tr>
<tr>
<td>102400</td>
<td>Gripper Driver Software Package</td>
</tr>
<tr>
<td>102500</td>
<td>Pressure Sensor Software Package</td>
</tr>
<tr>
<td>102600</td>
<td>Ultrasonic Sensor Software Package</td>
</tr>
<tr>
<td>102700</td>
<td>Camera System Software Package</td>
</tr>
<tr>
<td>103000</td>
<td>Frame Set</td>
</tr>
<tr>
<td>103100</td>
<td>Base Plate</td>
</tr>
<tr>
<td>103101</td>
<td>Robotic Arm Adapter</td>
</tr>
<tr>
<td>103102</td>
<td>Emergency Shut off</td>
</tr>
<tr>
<td>103200</td>
<td>Interfaces</td>
</tr>
<tr>
<td>103201</td>
<td>Handle Accessory</td>
</tr>
<tr>
<td>104000</td>
<td>Gripper</td>
</tr>
<tr>
<td>104100</td>
<td>End Point Gripper</td>
</tr>
<tr>
<td>104101</td>
<td>Palm</td>
</tr>
<tr>
<td>104102</td>
<td>Gear Link</td>
</tr>
<tr>
<td>104103</td>
<td>Link</td>
</tr>
<tr>
<td>104104</td>
<td>Finger</td>
</tr>
<tr>
<td>104105</td>
<td>Finger Tip Grip</td>
</tr>
<tr>
<td>104106</td>
<td>Screw</td>
</tr>
<tr>
<td>104200</td>
<td>Gripper Shell</td>
</tr>
<tr>
<td>104300</td>
<td>Motors</td>
</tr>
<tr>
<td>104301</td>
<td>Motor Gear</td>
</tr>
<tr>
<td>104302</td>
<td>Stepper Motor</td>
</tr>
<tr>
<td>104303</td>
<td>Stepper Motor Driver</td>
</tr>
<tr>
<td>104400</td>
<td>Wrist Assembly</td>
</tr>
<tr>
<td>105000</td>
<td>Sensors</td>
</tr>
<tr>
<td>105100</td>
<td>Force Sensor</td>
</tr>
<tr>
<td>105101</td>
<td>Bolsen Tech Pressure Sensor</td>
</tr>
<tr>
<td></td>
<td>105102</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>105200</td>
</tr>
<tr>
<td></td>
<td>105201</td>
</tr>
<tr>
<td></td>
<td>106000</td>
</tr>
<tr>
<td></td>
<td>106100</td>
</tr>
<tr>
<td></td>
<td>106101</td>
</tr>
<tr>
<td></td>
<td>106102</td>
</tr>
<tr>
<td></td>
<td>106103</td>
</tr>
<tr>
<td></td>
<td>106104</td>
</tr>
<tr>
<td></td>
<td>106105</td>
</tr>
<tr>
<td></td>
<td>107000</td>
</tr>
<tr>
<td></td>
<td>107100</td>
</tr>
<tr>
<td></td>
<td>107200</td>
</tr>
<tr>
<td></td>
<td>107300</td>
</tr>
<tr>
<td></td>
<td>107400</td>
</tr>
<tr>
<td></td>
<td>107500</td>
</tr>
<tr>
<td></td>
<td>108000</td>
</tr>
<tr>
<td></td>
<td>108100</td>
</tr>
<tr>
<td></td>
<td>108101</td>
</tr>
<tr>
<td></td>
<td>108102</td>
</tr>
<tr>
<td></td>
<td>109000</td>
</tr>
<tr>
<td></td>
<td>109100</td>
</tr>
</tbody>
</table>

*For all hardware drawings see IDL section above.*
Wire Diagrams:

**Raspberry Pi Pinout Diagram**

![Raspberry Pi 4 Pinout Diagram]

https://www.theengineeringprojects.com/2021/03/what-is-raspberry-pi-4-pinout-specs-projects-datasheet.html
This wiring diagram shows the circuit setup for the gripper motor. We have chosen a Nema 42 N-cm torque stepper motor because it gives our gripper better precision in movement and stronger grip strength when holding a product. The Raspberry Pi will act as provider for 3.3V source to power the motor driver, the 5V source to power the motor and the ground. The Raspberry Pi will also control the steps through GPIO 20 and control the direction of the motor through pin GPIO 21. A and B labeled pins are labeled to indicate connection to the two different inductors that control the motor (Inductor A and Inductor B) where the numbers 1 and 2 indicate the ends that they are connected to. The colors of these wire pins are equivalent to the wire colors of the stepper motor and are dotted to illustrate a difference to the other wires of similar color.
The diagram above depicts the circuit used for a single force sensor. Given that our gripper requires one for each fingertip, this circuit will be repeated twice but on different input nodes of the Raspberry Pi. This circuit utilizes an analog to digital (MCP3008) component to measure force from the information provided by the sensor. The circuit also utilizes the 3.3V output from the Raspberry Pi and a 1kΩ resistor to control the current from the force sensor. The larger the resistor the more sensitive the pressure sensor will be.
Raspberry Pi to Camera

<table>
<thead>
<tr>
<th>signal</th>
<th>I/F board</th>
<th>DC-DC</th>
<th>conn</th>
<th>wire size</th>
<th>wire color</th>
</tr>
</thead>
<tbody>
<tr>
<td>camera</td>
<td></td>
<td></td>
<td>1</td>
<td>15 pins</td>
<td>Blue ribbon</td>
</tr>
</tbody>
</table>

The wiring diagram for the camera is simply a 15-pin FPC ribbon cable connected to the camera module port.
The previous wire diagram displays the wiring configuration for the emergency stop button. When pressed it will return the gripper to an open position where it will stop motion. Since it was necessary to have 2 Raspberry Pi computers for this project, this button will be plugged into the camera system Raspberry Pi.

### I/F board to Conn Wiring List

<table>
<thead>
<tr>
<th>signal</th>
<th>I/F board</th>
<th>DC-DC</th>
<th>conn</th>
<th>wire size</th>
<th>wire color</th>
<th>PIN</th>
<th>GPIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>1</td>
<td>15</td>
<td>Blue</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>1</td>
<td>15</td>
<td>Black</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ultrasonic Sensor

<table>
<thead>
<tr>
<th>signal</th>
<th>I/F board</th>
<th>DC-DC</th>
<th>conn</th>
<th>wire size</th>
<th>wire color</th>
<th>PIN</th>
<th>GPIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td>Red</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>2</td>
<td>15</td>
<td>15</td>
<td>Black</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trig</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td>Green</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Echo</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td>Blue</td>
<td>11</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

The above wire diagram shows how the ultrasonic sensor is connected to the raspberry pi. It is important to note the inclusion of two resistors. The first, 1k ohm resistor, is between the echo pin and pin 11 on the board. The second, 2k ohm, goes from pin 11 to ground.
Software Design Document:

IEEE 1016 Software Design Description standard was used in the organization and content guidelines for the SDD of the Adaptive Robot Gripper

Software Design Document (SDD)

1 Scope

The document shall include an overview of the system that the software is part of, identification of the software configuration items, and a document overview.

The adaptive robot gripper will use software mainly in three different areas: 1) movement control, 2) force feedback, and 3) object recognition/location. To allow a mechanical gripper the ability to adapt and move any product, it must have the ability to recognize the product that it is picking up and understand the environment in which it operates. The system can gather this information which will allow it to adjust the gripper to specific products and move them from one desired location to a final desired location. The main inputs from which our software will gather information are the 1) camera, which is used for object recognition, 2) force resistors, which is used for force feedback, and 3) LIDAR, which is used for object location. The system uses the camera to visually identify various products through object recognition so it understands the product that it will interact with. The trained image algorithm will figure out how the specific product needs to be picked up from preprogramed information gathered for all the different products that the gripper will interact with. The force resistors will provide constant feedback as the gripper interacts with the product, and the control will decide whether to apply force as needed. The LIDAR can identify the location of the object relative to itself, and this information will be passed to the software to process and move the arm to the correct position. The main outputs are 1) stepper motor, which will open and close the gripper to the correct size depending on the size/weight/density of the object, and 2) robot arm, which will move the gripper to the correct location and orientation.

The purpose of this document is to show the intention of the project software, explain how the created system works, and to provide evidence and code that shows how it works. This document will go over the reference documents, CSCI-Wide design decisions, CSCI architectural design, CSCI detailed design, requirement traceability, and additional notes from the Adaptive Robot Gripper team. With the information from this document, any user should be able to understand how the software works without having to look at the actual code or watching the device function.

2 Referenced Documents

- Critical Design Element
- Project Fundamentals, Modeling, and Prototyping Plan
- Proposal
- System Block Architecture
- System Requirements Document
- System Requirements Flowdown
3 Software Specific Requirements and Traceability

All system requirements are connected to software.

2.1 Camera System:

2.1.1 Object recognition: Camera system shall detect and recognize the type of object to be gripped

Camera system shall use Raspberry Pi OS (64-bit) and C++ libcamera library to take pictures. The yolov5 object recognition was developed on Python but has been ported over to C++. The yolov5 bounding box machine learning software was developed on Python, but Raspberry Pi 64-bit only supports C++ with libcamera. The images were annotated using Roboflow.com, and trained into a .pt file, which was converted into a .onnx format. For our IDE, we used Visual Studio Code, and GitHub was used for version control.

2.1.2 Stationary: The camera system shall be stationary above the conveyer system & the ARG shall have a default location

The ARG default location must be stored in the code.

2.2 Sensors and software:

2.2.1 Data lookup: The Raspberry Pi shall retrieve all relevant data for each object

The relevant data for each object including height, proper grip orientation, and desired gripping force must be stored.

2.2.2 Distance Sensor: The ARG shall be able to detect the distance (x) from the object.

The ultrasonic sensor is connected to the Raspberry Pi and controlled by code.

2.2.3 Force Sensor: The ARG shall be able to apply the appropriate force to the object by having the ability to measure the force it is currently applying.

The force sensor is connected to the Raspberry Pi and data is read via the analog to digital converter. The code must properly apply physics mechanics to the gripping system.

2.2.4 Emulation: The ARG shall utilize emulation software in place of a robot arm for demonstration purposes

The emulation will be done using RoboDK program, which can be deployed on the Raspberry Pi, and use field data for emulation.

2.3 Performance and cost:

2.3.1 Inexpensive: The cost of the ARG shall not exceed $4000

All software is free, open source, or license has been provided by the University of Arizona. The only software that has a purchased license is the RoboDK application.

2.3.2 Performance: The ARG shall have a success rate greater than 99%.
The fully integrated software must work with the GUI and each subcomponent to produce a fully functioning conveyer system, free of bugs.

2.3.3 Pick rate: The ARG shall have a pick rate of at least 30 picks per minute

The camera recognizes each object before it reaches the gripping location, so it does not account for the time. However, the constant variables in the code can be changed to move the stepper motor faster when opening/closing the gripper in an actual factory. Code as is worked during demo day and was slowed down to allow a human operator to grip the object.

2.4 Mechanical and Design:

2.4.1 Adaptable: The ARG shall be able to grip all required Unilever products

The gripper was be designed using Solidworks CAD.

2.4.2 Handle: The ARG will have a handle for demonstration

The handle was be designed using Solidworks CAD.

2.4.3 Safety: The ARG will have a shutdown mechanism for emergency situations and failsafe protocol

The button and control software is integrated into the gripper software. Pressing the gripper will terminate the program.

4 Software Design

CSCI - Client Server Communications Interface

4.1 Software Wide Design Decisions

a. CSCI Inputs/Outputs
   The adaptive robot gripper should be able to function without any inputs from the client. The only necessary input is the failsafe switch. However, the ARG will need to be preprogrammed with a list of items to identify products as well as the best way to pick those items up. Outputs include the location of the product, the identification of the product, and the movement of the gripper to the correct location. There will be constant feedback between the force sensors and the gripper as it picks up the object and moves it, using software to determine if force needs to be applied to firmly grip the product but also not destroy it.

b. CSCI behavior in response to varying conditions, actions, algorithms, equations, handling of invalid inputs/conditions
   Open-source libraries will be used to feed in sample images so the algorithms can learn to identify products. One important invalid condition is that if the object recognition is not above 70% sure, the gripper will error and not do anything. This is for safety reasons, as we do not want an unidentified object being moved. Physics equations will be implemented to use the data from the force sensor to determine the correct force to apply to an object based, though preprogrammed grip locations and weight may be needed for simplicity and speed. All relevant equations and derivations are in the models section.
c. **User Interface**
   A user interface will be created so users can upload images and data of new products and manage existing products. Other UI may include displaying the images, location, and force sensor data.

d. **Safety, security, privacy**
   For safety, the gripper will not grip anything that it does not recognize with less than 70% confidence. Security and privacy include making sure the microcontroller is not connected to the internet during operation to avoid being hacked, and the camera should not store any images it takes besides the sample data.

e. **CSCI flexibility, availability, maintainability**
   The source code will be available for inspection and modification through a GitHub repository.

4.2 **Software Architectural Design Decisions**

a. **CSCI Component Identification**
   i) Raspberry Pi 4FSR 402 force sensitive resistors
   ii) Raspberry Pi HQ Camera Module
   iii) HC SR-04 ultrasonic sensors
   iv) MCP3008 Analog to Digital Converter
   v) Stepper Motor
   vi) Stop button

b. **Static relationship – N/A**
   c. **Purpose – N/A**
   d. **Status – N/A**
   e. **Planned computer resources used – N/A**
   f. **Software libraries**
      i) libcamera API
      ii) Yolov5 Convolutional Neural Network
      iii) RoboDK API
   g. **Concept of Execution**
      The ultrasonic sensor will detect when an object is close enough on the conveyer belt, and the camera module then feed an image of the object to the microcontroller. The microcontroller will use markers to determine the location of the object and algorithms to determine the product type. Preprogrammed weights and best grip location will be fed to the gripper, which will move accordingly to grip and transfer the product. After the object is successfully transferred, the gripper will then return to its original position, ready for another cycle. This all happens within 2 seconds. If an unknown object is detected, the gripper will do nothing.

g. **Interface Design**
   The graphical user interface displays the status of each step in the whole gripping system. There are 9 steps: 1) object detection, 2) object identified, 3) object located, 4) object grabbed, 5) object moving, 6) object dropped, 7) cycle complete. RED denotes not started, YELLOW denotes started, GREEN denotes completed. The GUI also displays the image taken with any bounding boxes around the object, as well as a confidence level, distance/location display, and force sensor readout.

4.3 **Software Detailed Design**
Full software can be found on the GitHub repository at https://github.com/UA-joealexliang/Team-22053_AdaptiveRobotGripper

Full software code can also be found in the appendix

5  Software Test Plan and Description

a. Software Test Plan
   The testing environment and location will be at a University of Arizona lab. The robotic gripper team will be present, and the Raspberry Pi 4 microcontroller, Camera, HC SR-04, and MCP3008 analog to digital converter was used for Test 1: Object Identification and Location. The additional robot gripper and FSR 402 sensors will be used for Test 2: Adaptive Gripping and Movement.

b. Software Test Description
   a. Test 1: Object Identification
      i. Functionality Test
      ii. Objective: ensure that the objects are being correctly identified
          Conditions: multiple products, as well as no product and unknown item
      iii. Data recording
          Record the object recognition result, and compare them to the real answers
      iv. Test Description
          The object recognition software has a threshold confidence of >70% for an object to be classified as a specific product. A test constitutes a FAIL if a product is identified as another product, a non-product is identified as a product, or a product is not identified. A test constitutes a PASS if none of these conditions are violated.
      v. Requirements
          The whole test constitutes a PASS if all 5 products are each correctly identified three times, and no conditions are violated. This meets our overall threshold of 100% success.
   b. Test 1: Object Location
      i. Functionality Test
      ii. Objective: ensure that the HC SR-04 ultrasonic sensor can properly measure distance within +/-5mm.
          Conditions: 5 varying distances
      iii. Data recording
          Record the object distance result, and compare them to the real answers
      iv. Test Description
          Use the software to interface with the ultrasonic sensor, which outputs a distance value when an object is in front of the sensor. Compare this value to the actual measured value. A test constitutes a PASS if the measured data from the ultrasonic sensor is with +/-5mm tolerance of the actual value.
      v. Requirements
          The whole test constitutes a PASS if all 5 different locations are within +/-5mm.
   c. Test 2: Force Sensor Calibration
      i. Functionality Test
ii. Objective: ensure that the force sensors are properly calibrated and displaying the correct force applied in Newtons

iii. Data recording
Record the output from the force sensor analog to digital converter.

iv. Test Description
The test will be placing products of known weight on the resistor uniformly, and then seeing if the measured force data is +/-1N of the actual value. A PASS meets these requirements.

v. Requirements
The whole test constitutes a PASS if the data output meets the error threshold of three varying forces applied to the resistor.

d. Test 2: Adaptive Gripping and Movement
i. Functionality Test

ii. Objective: ensure that the objects are being gripped properly by the gripper
Conditions: multiple products, as well as no product and unknown item

iii. Data recording
Record the gripping process, and look for any issues

iv. Test Description
A test constitutes a PASS if no products are crushed or slipping, foreign objects are not picked up or the failsafe switch activation works properly.

v. Requirements
The whole test constitutes a PASS if all 5 products are properly handled.

6 Software Version Description

a. Software
i. SolidWorks CAD – used to model the adaptive robot gripper
Solidworks is a computer-aided design software; its license is provided by the University of Arizona for use in modeling.

ii. Raspberry Pi OS – used to program the Raspberry Pi 4 microcontroller
Raspberry Pi OS is the operating system which the Raspberry Pi microcontroller will run on.

iii. Visual Studio Code IDE – coding layout environment
Visual Studio Code is an integrated development environment that is very popular for developers.

iv. GitHub – used to maintain and collaboratively program code
GitHub hosts software development and version control using Git.

v. RoboDK
RoboDK is a software we are using to provide emulation for the robot arm. It will run on the Raspberry Pi as information is fed to it when the camera system reacts to a product on the assembly line.

b. Source Code
https://github.com/UA-joealexliang/Team-22053_AdaptiveRobotGripper

c. External Libraries
i. yolov5 CNN to train models, one is original source code, another is ported to C++ for deployment on Raspberry Pi
   - https://github.com/ultralytics/yolov5
   - https://github.com/doleron/yolov5-opencv-cpp-python

ii. libcamera – this is the library that the Raspberry Pi HQ camera will use to perform actions such as taking pictures

7 Software User’s Manual

All the links below are all the resources the ARG team started with. They go into much more detail about specific components. This TDP will cover setting up everything on the Raspberry Pi, and how to train a neural network. Another separate manual has been made for ROBODK emulation. Ultrasonic sensor documentation, force sensor documentation, and stepper motor documentation can be found at the links below. As these components are not specific to our project, they will not be covered in the software user’s manual. We do have a general graph overview of how the different functions in our software all connect to each other.

a. SolidWorks CAD
   i. Download Link: https://softwarelicense.arizona.edu/solidworks

b. Raspberry Pi OS
   i. Download Link: https://www.raspberrypi.org/%20downloads/
   ii. How to build projects with Python: https://realpython.com/python-raspberry-pi/
   iii. Ultrasonic sensor documentation: https://www.electronicshub.org/raspberry-pi-ultrasonic-sensor-interface-tutorial/
   iv. Force sensor documentation: https://pimylifeup.com/raspberry-pi-pressure-pad/

c. Visual Studio Code IDE
   i. Download Link: https://code.visualstudio.com/download

d. GitHub
   i. Website Link: https://github.com/

e. Roboflow
   i. Website Link: https://roboflow.com/

f. RoboDK
   i. Website Link: https://robodk.com/
Raspberry Pi Basics

Raspberry Pi official documentation can be accessed at the link below:

Introduction

Unlike an Arduino (which you can upload code and it’ll automatically run), the Raspberry Pi is a fully functional operating system (OS). Think of it like a minicomputer with GPIO pins. You will need a (1) monitor, (2) keyboard, and (3) mouse to interface with the Raspberry Pi. The Raspberry Pi has USB ports, an HDMI port, and a SD card which houses the memory storage. The Raspberry Pi also has WiFi capabilities, which can be useful for remote access.

Troubleshooting: The Raspberry Pi is not displaying on the monitor

If there are problems getting the Raspberry Pi to display its home screen, turn off the Raspberry Pi and then turn it back on. The Raspberry Pi only seems to display if it is connected to the monitor first and then booted up. Plugging in the Raspberry Pi to a monitor after it has booted up may not work.

Which model of Raspberry Pi should I get?

We determined that a Raspberry Pi 4 was necessary for our work with the adaptive robot gripper. While some models are cheaper, they do not have Wifi, HDMI ports, or no place to insert a camera. Below are some links on comparing the differences between each model:

- https://opensource.com/life/16/10/which-raspberry-pi-should-you-choose-your-project

Installing the Operating System (Bullseye 64-bit operating system)

The operating system for the Raspberry Pi must be installed on the SD card through the Raspberry Pi Imager on your computer. Make sure you have a SD card reader on your computer or have a SD card to USB reader. The download for Raspberry Pi Imager can be found at this
link: https://www.raspberrypi.com/software/. Once you have the SD card connected to your computer, open the Raspberry Pi Imager and choose the Raspberry Pi OS (64-bit) Bullseye system to install. While the 32-bit Debian OS is more developed, there are certain libraries for our image recognition software that can only run on 64-bit operating systems. Although optional, it is recommended to also go into settings and choose a unique username and password for the Raspberry Pi, which will be useful when remote connecting via VNC Viewer. Our current Raspberry Pi 4 has the username ‘pi’ and the password ‘22053’.

How do I transfer files or set up remote access between my computer and the Raspberry Pi? (VNC Viewer)

Most of the adaptive gripper project software required editing/training/testing on a computer which has a stronger CPU and then deploying onto the Raspberry Pi. Transferring files can be done via a USB flash drive or using the already installed VNC Viewer. The VNC Viewer program is a software that allows for remote access via Wifi.

The download link for the VNC Viewer program for your computer can be found here: https://www.realvnc.com/en/connect/download/viewer/

Once installed, make sure both your Raspberry Pi and your computer are connected to the same WiFi network.
Open the VNC Viewer on the Raspberry Pi to get the correct server address. Enter this onto your computer’s VNC Viewer and enter the username and password for the Raspberry Pi. The default username is ‘raspberry’ and the default password is ‘pi’, though it is recommended you change this information during the installation process. Once a connection has been successfully established, you will be able to remotely access the Raspberry Pi from your computer.

**What Python libraries need to be installed on the Raspberry Pi? (Doleron, OpenCV)**

Link to YOLOv5 official repository:
- [https://github.com/ultralytics/yolov5](https://github.com/ultralytics/yolov5)

Link to Doleron’s YOLOv5 OpenCV implementation:
- [https://github.com/doleron/yolov5-opencv-cpp-python](https://github.com/doleron/yolov5-opencv-cpp-python)

Link to documentation on how to install OpenCV 4.5:
- [https://qengineering.eu/install-opencv-4.5-on-raspberry-64-os.html](https://qengineering.eu/install-opencv-4.5-on-raspberry-64-os.html)

Due to the Raspberry Pi using a custom Linux OS, there are many dependencies that make training the YOLOv5 neural network unfeasible on the Raspberry Pi itself. The image recognition software should be trained on a computer, and the exported .onnx file will be the only thing that needs to be sent over to the Raspberry Pi. Since the official YOLOv5 software has many libraries that cannot be installed on the Raspberry Pi, we also use Doleron’s OpenCV image recognition which only requires (1) modern Linux OS, (2) OpenCV 4.5.4+, and (3) Python 3.7+. The Raspberry Pi is a modern Linux OS and has Python 3.7+ already installed, so the only thing that needs to be installed is OpenCV 4.5.4+.

**How to install OpenCV 4.5.4+**

This is a shortened version of the Q-engineering manual.

1. Run the commands ‘uname -a’ and ‘gcc -v’ to ensure that the Raspberry Pi is aarch64-linux-gnu C++ compiler. We did encounter any problems when doing this with the Raspberry Pi 4.
2. Ensure you have enough RAM. If you have a Raspberry Pi with 4 or 8 GB of memory, it is sufficient to install OpenCV. If you only have 2 GB of RAM, follow the screenshot below.

```bash
$ sudo nano /usr/bin/zram.sh
# alter the limit with * 2
mem=$(( ($totalmem / $cores) * 1024 * 3))
# save with <Ctrl+X>, <Y> and <Enter>
$ sudo reboot
```

[Image of the screenshot showing the code and the output of `free -m` command]

```
pi@raspberrypi:~ $ free -m
Mem: 1798 207 1334 38 256 1493
Swap: 5394
```

1.7 + 5.4 = 7.1 GByte only needed for version 4.5.2
3. Installing OpenCV is not complicated, but it will take about one and a half hour to complete.

The following code utilizes Q-Engineering’s installation script:

```
# check your memory first
$ free -m
# you need at least a total of 6.5 GB!
# if not, enlarge your swap space as explained in the guide
$ wget https://github.com/Qengineering/Install-OpenCV-Raspberry-Pi-64-bits/raw/main/OpenCV-4-5-5.sh
$ sudo chmod 755 ./OpenCV-4-5-5.sh
$ ./OpenCV-4-5-5.sh

If you prefer to manually install the library, follow the code below:

First, we need to install other third-party software libraries that OpenCV uses.

```
# check for updates (64-bit OS is still under development!)
$ sudo apt-get update
$ sudo apt-get upgrade
# dependencies
$ sudo apt-get install build-essential cmake git unzip pkg-config
$ sudo apt-get install libjpeg-dev libpng-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
$ sudo apt-get install libgtk2.0-dev libcanberra-gtk* libgtk-3-dev
$ sudo apt-get install libgstreamer1.0-dev gstreamer1.0-gtk3
$ sudo apt-get install libgstreamer-plugins-base1.0-dev gstreamer1.0-gl
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install python3-dev python3-numpy python3-pip
$ sudo apt-get install liblapack-dev gfortran libhdf5-dev
$ sudo apt-get install libprotobuf-dev libgoogle-glog-dev libgflags-dev
$ sudo apt-get install protobuf-compiler
```

Then, download OpenCV 4.5.5 using the code below:

```
# check your memory first
$ free -m
# you need at least a total of 6.5 GB!
# if not, enlarge your swap space as explained earlier
# download the latest version
$ cd ~
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.5.zip
```
$ wget -O opencv_contrib.zip
https://github.com/opencv/opencv_contrib/archive/4.5.5.zip
# unpack
$ unzip opencv.zip
$ unzip opencv_contrib.zip
# some administration to make live easier later on
$ mv opencv-4.5.5 opencv
$ mv opencv_contrib-4.5.5 opencv_contrib
# clean up the zip files
$ rm opencv.zip
$ rm opencv_contrib.zip

4. Build Make. You have to make a directory where all the build files can be located

$ cd ~/opencv
$ mkdir build
$ cd build

Here you tell CMake what, where, and how to make OpenCV on your Raspberry.

$ cmake -D CMAKE_BUILD_TYPE=RELEASE \ 
   -D CMAKE_INSTALL_PREFIX=/usr/local \ 
   -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \ 
   -D ENABLE_Neon=ON \ 
   -D WITH_OPENMP=ON \ 
   -D WITH_OPENCL=OFF \ 
   -D BUILD_TIFF=ON \ 
   -D WITH_FFMPEG=ON \ 
   -D WITH_TBB=ON \ 
   -D BUILD_TBB=ON \ 
   -D BUILD_GSTREAMER=ON \ 
   -D BUILD_TESTS=OFF \ 
   -D WITH_Eigen=OFF \ 
   -D WITH_V4L=ON \ 
   -D WITH_LIBV4L=ON \ 
   -D WITH_VTK=OFF \ 
   -D WITH_QT=OFF \ 
   -D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages \ 
   -D OPENCV_GENERATE_PKGCONFIG=ON \ 
   -D BUILD_EXAMPLES=OFF ..
Hopefully, everything went well and CMake comes out with a report that looks something like the screenshot below.
5. Make. With all compilation directives in place, you can start the build with the following command. This will take a while (minimum 1hr30min). The following shows what a successful build will look like.

To complete, install all the generated packages to the database.

```
$ sudo make install
$ sudo ldconfig
cleaning (frees 300 KB)
$ make clean
$ sudo apt-get update
```
6. Checking. To check your installation, try typing in the following commands.

```bash
pi@raspberrypi:~/opencv/build $ python3
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.5.2'
>>> 
pi@raspberrypi:~/opencv/build $ 
```
How To Train A YOLOv5 Neural Network

Introduction

Link to YOLOv5 GitHub Repository

- https://github.com/ultralytics/yolov5

The YOLOv5 neural network model is used specifically for image recognition machine learning. The one we used for this project was specifically the bounding box image recognition, since it can draw a box around the object. This allows for easier debugging, greater accuracy, and the ability to identify when there is nothing in the picture. The following manual will be a thumbed down and simplified version of the YOLOv5 instructions, tailored specifically to our ARG project.

Gathering Images For Training

When gathering images for training, the best way to get a good model is to have a variety of images of the object. Try taking pictures of the object from different angles, different distances, under different lighting, and rotate the object to capture all its sides. However, do make sure that the whole object is in the frame of the picture, or else the model will not train well.

If you are planning to capture images using an iPhone, make sure to go into Settings>Camera>Formats and change to ‘Most Compatible’, as this will ensure the pictures save as JPEG files. Using ‘High Efficiency’ will save the images as HEIF/HEVC files, which YOLOv5 cannot use to train the model.

There are many apps in the App Store that have timers to take pictures quickly every x seconds, so you will not have to click the snap picture button every time. The one our team used was
‘Lens Buddy’. Make sure you change the grid type to square, or else the cropping will mess up during the Roboflow annotation. YOLOv5 only works with square images.

How do we draw the bounding boxes and export data for training? (Roboflow)

Since we are using bounding boxes, there is a little more preprocessing to do. We need to manually draw all the bounding boxes for all the different objects, so the model knows where to look when training. One popular online website to do this is Roboflow.

1. Create a new dataset and import all the images you’ve taken. Make sure to set the project type to ‘Bounding Box’.
2. Label all the products in the images.
3. Once all the images have been labeled, go ahead and export. There are a few settings, the only ones to look out for is set the image size to 640x640 pixels. The larger the pixel count the more detailed the image will be, but the longer it will take for the model to train. You can apply augmentations such as lighting and exposure to the images as well, but YOLOv5 already does that and there is a limit to how many images you can export for free on Roboflow.
Generating New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

1. Source Images
   - Images: 222
   - Classes: 5
   - Unannotated: 0

2. Train/Test Split
   Here’s how you split your images when you added them to the dataset:
   - Training Set: 159 images
   - Validation Set: 42 images
   - Testing Set: 21 images

3. Preprocessing
   - Decrease training time and increase performance by applying image transformations to all images in this dataset.

4. Augmentation
Generate New Version

Train/Test Split
Validation Set: 42 Images
Testing Set: 21 Images

Preprocessing
Auto-Orient: Applied
Resize: Stretch to 640x640

Augmentation
Create new training examples for your model to learn from by generating augmented versions of each image in your training set.
Add Augmentation Step
Continue
Generate

Unannotated: 0

Train/Test Split
Training Set: 159 Images
Validation Set: 42 Images
Testing Set: 21 Images

Preprocessing
Auto-Orient: Applied
Resize: Stretch to 640x640

Augmentation
Turned Off

Generate

Review your selections then click “Generate” to create a moment-in-time snapshot of your dataset with the applied preprocessing steps.

Maximum Version Size: 222
See how this is calculated ->
Generate
How to train the YOLOv5 model

First, git clone the repository from the link. Then, place all your exported image data into the file.

The command below can be entered into the Visual Studio Code terminal to train your own specific dataset. A batch of 16 and 300 epochs is recommended for your local machine. Larger batches need more RAM to train.

$ python path/to/train.py --data unilever.yaml --weights yolov5s.pt --batch 16 --epochs 300 --img 640

In the unilever.yaml file, you’ll need to have the path for your images, as well as the different product names.

```yaml
path: ../unilever-products-yaml  # dataset root dir
train: train/images
val: valid/images
#test: test/images
nc: 5
names: ['bodywash', 'deodorant', 'soap', 'spray', 'tea']
```

How to export the YOLOv5 model

The YOLOv5 model comes out as a .pt file. Doleron’s Raspberry Pi compatible implementation uses .onnx. Use the following command to export the model as a .onnx model.

```bash
python export.py --weights bestexp16.pt --include onnx
```
How to set up the model on Doleron’s Repository

First, git clone Doleron’s repository. Then go into python folder and place the full integrated software code in there. Then, place your .onnx model into the config_files folder, with the classes .txt document.
Your program is now ready to run.
8 Appendix

Full Software Program

```python
# Libraries
import tkinter as tk
from tkinter import ttk
from PIL import ImageTk, Image
import numpy as np
import matplotlib.pyplot as plt  # unused, supposed to graph force sensor
import os
import time
import threading
import subprocess

# GUI Libraries
import threading
import sys
import os
import RPi.GPIO as GPIO  # Import Raspberry Pi GPIO library
import time

# Button Libraries, Image Recognition Libraries, Distance Sensor Libraries, Stepper Motor
import threading
import sys
import os
import RPi.GPIO as GPIO  # Import Raspberry Pi GPIO library
import time

# Image Recognition Libraries
import cv2
import numpy as np
import subprocess

# Pressure Sensor Libraries
import spidev

# Global Variables
is_new_image = False  # sets to true when a new image is taken, and identified
path_to_img = 'yolov5.png'
img_resize = (320, 320)
```
PressureB = None #recorded pressure in bits
PressureN = None #recorded pressure in Newtons
Actual_Pressure = None
Distance = None
Guess = None #object identified
Confidence = None #confidence of object identified
Object = None #do not modify, string concentration of Guess and Confidence
CONFIDENCE_THRESHOLD = 0.4 #what confidence does a recognized object need to pass?

#[Stepper Motor Constants]#
DIR = 20 #Direction GPIO Pin
STEP = 21 #Step GPIO PIN
CW = 0 #CLOCKWISE ROTATION
CCW = 1 #CCW ROTATION
SPR = 800 #steps per revolution
GPIO.setmode(GPIO.BCM)
GPIO.setup(DIR,GPIO.OUT)
GPIO.setup(STEP,GPIO.OUT)
GPIO.output(DIR,CW)
MODE = (14,15,18) #MICROSTEP RESOLUTION GPIO
GPIO.setup(MODE,GPIO.OUT)
RESOLUTION = {'1':(0,0,0), #Full
              '2':(1,0,0), #Half
              '4':(0,1,0), #1/4
              '8':(1,1,0), #1/8
              '16':(0,0,1), #1/16
              '32':(1,0,1)} #1/32

#[Distance Sensor Constants]#
GET_DISTANCE_DELAY = 0.05 #delay between each distance sensor read
PRINT_DISTANCE = False
DISTANCE_SENSOR_LOADED = False
DISTANCE_THRESHOLD = 18 #18 cm was measured to be the width of the conveyor belt

#[Force Sensor Constants]#
GET_FORCE_DELAY = 0.1 #delay between each force sensor read
PRINT_FORCE = True
pad_channel = 0
measured_weight = 10
measured_bit = 1023
measured_n = measured_bit/(measured_weight*9.807) #max value is 1023 bits, corresponding to 10kg
#Create SPI
spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz=1000000
stop_forceThread = False

# [Image Recognition Constants]#
INPUT_WIDTH = 640
INPUT_HEIGHT = 640
SCORE_THRESHOLD = 0.2
NMS_THRESHOLD = 0.4
CONFIDENCE_THRESHOLD = 0.4

GUI Software

border_effects = {
    "flat": tk.FLAT,
    "sunken": tk.SUNKEN,
    "raised": tk.RAISED,
    "groove": tk.GROOVE,
    "ridge": tk.RIDGE,
}

# sets up window
window = tk.Tk()
window.title('ARG GUI - 22053')
window.geometry("1920x1080")

# command to destroy the GUI window
def end():
    window.destroy()

# takes whatever image from folder as output from camera system
PIL_image = Image.open(path_to_img)
resize_image = PIL_image.resize(img_resize)
img = ImageTk.PhotoImage(resize_image)
Scan = tk.Label(image=img)

# this outlines the array for steps and their values
Overview = [[0 for j in range(2)] for i in range(7)]
Overview[0][0]="Object Detection"
Overview[1][0]="Object Identified"
Overview[2][0]="Object Located"
Overview[3][0]="Object Grabbed"
Overview[4][0]="Object Moving"
Overview[5][0]="Object Dropped"
Overview[6][0]="Cycle Complete"
#Sets column 2 all values to 0
for i in range(0,7):
    Overview[i][1] = 0

#If statement for object confidence
def ConCheck():
    global Object
    if Confidence is None:
        Object = "Object Identity: Waiting for Data"
    elif Confidence > CONFIDENCE_THRESHOLD:
        Object = "Object: " + Guess + " Confidence: " + str(Confidence) + " Distance: "+ str(Distance) + " cm"
    else:
        Object = "The object could not be Identified"
    Identity_value.config(text = Object)

#pressure sensor GUI
def PressUpdate():
    if (PressureB is None or PressureN is None):
        Pressure_Value.config(text = "Pressure Value: Waiting for Data")
    else:
        Pressure_Value.config(text = str(PressureB) + " bits " + str(PressureN) + " Newtons")

#This section defines all labels for steps of ARG
Object_label = tk.Label(window, text = Overview[0][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Identified_label = tk.Label(window, text = Overview[1][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Located_label = tk.Label(window, text = Overview[2][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Grabbed_label = tk.Label(window, text = Overview[3][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Moving_label = tk.Label(window, text = Overview[4][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Dropped_label = tk.Label(window, text = Overview[5][0], font=('calibre',10, 'bold'), relief="raised", width=50)
Done_label = tk.Label(window, text = 'Overview[6][0]', font=('calibre',10, 'bold'), relief="raised", width=50)
Camera_label = tk.Label(window, text = "Camera System", font=('calibre',10, 'bold'), relief="raised", width=50)
Pressure_label = tk.Label(window, text = "Pressure System", font=('calibre',10, 'bold'), relief="raised", width=50)
Identity_label = tk.Label(window, text = "Object Identity", font=('calibre',10, 'bold'), relief="raised", width=50)

#This sets up the frames for the status system
#manually lays out the 7 steps on the grid
Object_label.grid(row=0,column=0, sticky="w")
Identified_label.grid(row=1,column=0, sticky="w")
Located_label.grid(row=2,column=0, sticky="w")
Grabbed_label.grid(row=3,column=0, sticky="w")
Moving_label.grid(row=4,column=0, sticky="w")
Dropped_label.grid(row=5,column=0, sticky="w")
Done_label.grid(row=6,column=0, sticky="w")

#Prototype of the status system using numbers
Object_value = tk.Label(window, width=10, bg="red")
Identified_value = tk.Label(window, width=10, bg="red")
Located_value = tk.Label(window, width=10, bg="red")
Grabbed_value = tk.Label(window, width=10, bg="red")
Moving_value = tk.Label(window, width=10, bg="red")
Dropped_value = tk.Label(window, width=10, bg="red")
Done_value = tk.Label(window, width=10, bg="red")
Pressure_Value = tk.Label(window, text = "Pressure Value: Waiting for Data" , font=('calibre',10, 'bold'), relief="raised", width=50)
Identity_value = tk.Label(window, text = "Object Identity: Waiting for Data", font=('calibre',10, 'bold'), relief="raised", width=50)
P_v = [0 for i in range (9)]
P_v[0] = Object_value
P_v[1] = Identified_value
P_v[2] = Located_value
P_v[3] = Grabbed_value
P_v[5] = Dropped_value
P_v[6] = Done_value
P_v[7] = Pressure_Value
P_v[8] = Identity_value

#Manually lays out all 7 values
Object_value.grid(row=0,column=1, sticky="w")
Identified_value.grid(row=1,column=1, sticky="w")
Located_value.grid(row=2,column=1, sticky="w")
Grabbed_value.grid(row=3,column=1, sticky="w")
Moving_value.grid(row=4, column=1, sticky="w")
Dropped_value.grid(row=5, column=1, sticky="w")
Done_value.grid(row=6, column=1, sticky="w")

# Creates command to change status values which should lead into color changes if possible

def set(n, val):
    Overview[n][1] = val

def check(n):
    if Overview[n][1] == 0:
        P_v[n].config(bg="red")
    elif Overview[n][1] == 1:
        P_v[n].config(bg="yellow")
    elif Overview[n][1] == 2:
        P_v[n].config(bg="green")

def reset():
    global PressureB, PressureN, Actual_Pressure, Guess, Confidence, Distance
    PressureB = None
    PressureN = None
    Actual_Pressure = None
    Guess = None
    Confidence = None
    Distance = None
    # ConCheck()
    # PressUpdate()
    for i in range(0, 7):
        Overview[i][1] = 0
        # P_v[i].config(bg="red")
        # print("red")

exit_button = ttk.Button(window, text="Exit", command=end)
reset_button = ttk.Button(window, text="Reset", command=reset)

# Displays label and image from camera system
Camera_label.grid(row=0, column=3, sticky="s")
Scan.grid(row=1, column=3, rowspan = 6)

# Displays Pressure values and label
Pressure_label.grid(row=10, column=3)
Pressure_Value.grid(row=11, column=3)

# Displays Object identity and confidence
Identity_label.grid(row=10, column=0)
Identity_value.grid(row=11, column=0)

# Displays graph and exit button
exit_button.grid(row=13)
reset_button.grid(row=12)

def GUI_Thread():
    while True:
        window.update()
        window.update_idletasks()
        ConCheck()
        PressUpdate()
        for i in range(0, 7):
            check(i)
            global is_new_image
        if is_new_image == True:
            is_new_image = False
            PIL_image = Image.open(path_to_img)
            resize_image = PIL_image.resize(img_resize)
            img = ImageTk.PhotoImage(resize_image)
            Scan = tk.Label(image=img)
            Scan.grid(row=1, column=3, rowspan=6)

# guiThread = Thread(target=GUI_Thread)
# guiThread.start()

#############################################################################
#############################

'''
Button Software
'''

class MyThread(threading.Thread):
    def run(self):
        print('Hello and good bye.')
        os._exit(1) # ends program in case of emergency

num_button_pushed = 0

def button_callback(channel):
    global num_button_pushed
    num_button_pushed = num_button_pushed + 1
    print("Button was pushed! ", num_button_pushed)
    MyThread().start()
def open_steps(steps=1600, delay=.005/16):
    print(steps)
    GPIO.output(MODE, RESOLUTION['16'])
    GPIO.output(DIR, CW)
    for x in range(steps):
        GPIO.output(STEP, GPIO.HIGH)
        sleep(delay)
        GPIO.output(STEP, GPIO.LOW)
        sleep(delay)

def close_steps(steps, delay):
    GPIO.output(DIR, CCW)
    for x in range(steps):
        GPIO.output(STEP, GPIO.HIGH)
        sleep(delay)
        GPIO.output(STEP, GPIO.LOW)
        sleep(delay)

def reset_close(steps, delay=0.005/16):
    GPIO.output(MODE, RESOLUTION['16'])
    GPIO.output(DIR, CCW)
    for x in range(steps):
        GPIO.output(STEP, GPIO.HIGH)
        sleep(delay)
        GPIO.output(STEP, GPIO.LOW)
        sleep(delay)

def grip_init():
    INIT_RES = '16'
    GPIO.output(MODE, RESOLUTION[INIT_RES])
    INIT_OPEN_STEPCNT = 1600
    INIT_OPEN_DELAY = 0.0005
    open_steps(INIT_OPEN_STEPCNT, INIT_OPEN_DELAY/INIT_RES)

def grip_bodywash():
    BODYWASH_RES = '16'
    GPIO.output(MODE, RESOLUTION[BODYWASH_RES])
    BODYWASH_STEPCNT = 1600
    BODYWASH_DELAY = 0.005/16
    close_steps(BODYWASH_STEPCNT, BODYWASH_DELAY)
    close_steps(200, .075/16)
    return 400, 1300
def grip_deodorant():
    DEODORANT_RES = '16'
    GPIO.output(MODE,RESOLUTION[DEODORANT_RES])
    DEODORANT_STEPCNT = 800
    DEODORANT_DELAY = 0.005/16
    close_steps(DEODORANT_STEPCNT, DEODORANT_DELAY)
    close_steps(1600, .005/16)
    return 400, 800

def grip_soap():
    SOAP_RES = '8'
    GPIO.output(MODE,RESOLUTION[SOAP_RES])
    SOAP_STEPCNT = 800
    SOAP_DELAY = 0.005/8
    close_steps(SOAP_STEPCNT, SOAP_DELAY)
    #close_steps(100,.075/16)
    return 400, 800

def grip_spray():
    SPRAY_RES = '8'
    GPIO.output(MODE,RESOLUTION[SPRAY_RES])
    SPRAY_STEPCNT = 850
    SPRAY_DELAY = 0.005/8
    close_steps(SPRAY_STEPCNT, SPRAY_DELAY)
    #close_steps(100,.075/8)
    return 400, 800

def grip_tea():
    TEA_RES = '16'
    GPIO.output(MODE,RESOLUTION[TEA_RES])
    TEA_STEPCNT = 1800
    TEA_DELAY = 0.005/16
    close_steps(TEA_STEPCNT, TEA_DELAY)
    return 400, 800

#https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range
#sensor-on-the-raspberry-pi
def distance_sensor():
    #first time? set everything up
global DISTANCE_SENSOR_LOADED
if DISTANCE_SENSOR_LOADED == False:
    GPIO.setmode(GPIO.BCM) #set reading pin to using bcm
    global PIN_TRIGGER
    PIN_TRIGGER = 4
    global PIN_ECHO
    PIN_ECHO = 17
    GPIO.setup(PIN_TRIGGER, GPIO.OUT) #out - triggers sensor
    GPIO.setup(PIN_ECHO, GPIO.IN) #in - reads return signal

    DISTANCE_SENSOR_LOADED = True
    print("DISTANCE_SENSOR_LOADED = True")

    #10us pulse to trigger the module
    GPIO.output(PIN_TRIGGER,GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(PIN_TRIGGER, GPIO.LOW)

    #echo pulse is low until called, then high for duration of pulse
    while GPIO.input(PIN_ECHO)==0:
        pulse_start_time = time.time()
    while GPIO.input(PIN_ECHO)==1:
        pulse_end_time = time.time()

    #math calculations
    pulse_duration = pulse_end_time - pulse_start_time
    distance = round(pulse_duration * 17150, 2)
    #alpha_distance = distance/2.54
    #print("Distance:",distance,"cm")
    #print("Distance:",alpha_distance,"in")
    return distance

def poll_distance_until(delay=GET_DISTANCE_DELAY, printD=PRINT_DISTANCE, threshold=DISTANCE_THRESHOLD):
    threshold_reached = False
    while threshold_reached==False:
        distance = distance_sensor()
        if printD==True:
            print("Distance:",distance,"cm")
        time.sleep(delay)
        if distance <= threshold:
            threshold_reached = True
            print("Object detected/located: ",distance,"cm")
        return distance
def poll_distance(delay=GET_DISTANCE_DELAY, printD=PRINT_DISTANCE):
    try:
        while True:
            distance = distance_sensor()
            if printD==True:
                print("Distance:",distance,"cm")
            time.sleep(delay)
    except KeyboardInterrupt:
        print("get_distance stopped by user")
        global DISTANCE_SENSOR_LOADED
        DISTANCE_SENSOR_LOADED = False
        print("DISTANCE_SENSOR_LOADED = False")
        GPIO.cleanup()

#################################################################################
#############################
'''
Force Sensor Software
'''

def readadc(adcnum):
    # read SPI data from the MCP3008, 8 channels in total
    if adcnum > 7 or adcnum < 0:
        return -1
    r = spi.xfer2([1, 8 + adcnum << 4, 0])
    data = ((r[1] & 3) << 8) + r[2]
    return data

def force_sensor():
    pad_value = readadc(pad_channel)
    measured_F = pad_value/measured_n
    return measured_F, pad_value

def poll_force(delay=GET_FORCE_DELAY, print_force=PRINT_FORCE):
    #try:
    print("Force Sensor on")
    while True:
        force, pad_value = force_sensor()
        global PressureB, PressureN
        PressureN = force
        PressureB = pad_value
        if print_force==True:
            print("Pressure Pad Value: %d" % pad_value)
            #print("Pressure Pad Force: %f" % force)
        time.sleep(delay)
if stop_forceThread==True:
    break
#except KeyboardInterrupt:
    # pass

Image Recognition Software

def remove_picture(path):
    if os.path.exists(path):
        os.remove(path)
    else:
        print("The file does not exist")

def take_picture():
    subprocess.run(["ls", "-l"])
    subprocess.run(["libcamera-jpeg","-o test.jpg", ",n", ",t", "1", ",--width", "640", ",--height", "640"])
    #save as ' test.jpg', 'no preview', '1 ms timeout'

#read the onnx trained model
def build_model(is_cuda):
    net = cv2.dnn.readNet("config_files/bestexp16.onnx")
    if is_cuda:
        print("Attempty to use CUDA")
        net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
        net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)
    else:
        print("Running on CPU")
        net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
        net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)
    return net

def detect(image, net):
    blob = cv2.dnn.blobFromImage(image, 1/255.0, (INPUT_WIDTH, INPUT_HEIGHT), swa
pRB=True, crop=False)
    net.setInput(blob)
    preds = net.forward()
    return preds

#load capture, returns cv2::Video
def load_capture():
capture = cv2.VideoCapture("sample.mp4")
return capture

def load_img(path):
    img = cv2.imread(path)
    return img

#load Unilever classes
def load_classes():
    class_list = []
    with open("config_files/unileverclasses.txt", "r") as f:
        class_list = [cname.strip() for cname in f.readlines()]
    return class_list

class_list = load_classes()

#for input_image, returns result_class_ids, result_confidences, result_boxes
def wrap_detection(input_image, output_data):
    class_ids = []
    confidences = []
    boxes = []

    rows = output_data.shape[0]

    image_width, image_height, _ = input_image.shape

    x_factor = image_width / INPUT_WIDTH
    y_factor = image_height / INPUT_HEIGHT

    for r in range(rows):
        row = output_data[r]
        confidence = row[4]
        if confidence >= CONFIDENCE_THRESHOLD: #change confidence level depending on lighting conditions
            classes_scores = row[5:]
            __, __, __, max_indx = cv2.minMaxLoc(classes_scores)
            class_id = max_indx[1]
            if (classes_scores[class_id] > .25):
                confidences.append(confidence)
                class_ids.append(class_id)
x, y, w, h = row[0].item(), row[1].item(), row[2].item(), row[3].item()

left = int((x - 0.5 * w) * x_factor)
top = int((y - 0.5 * h) * y_factor)
width = int(w * x_factor)
height = int(h * y_factor)
box = np.array([left, top, width, height])

boxes.append(box)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.25, 0.45)

result_class_ids = []
result_confidences = []
result_boxes = []

for i in indexes:
    result_confidences.append(confidences[i])
    result_class_ids.append(class_ids[i])
    result_boxes.append(boxes[i])

return result_class_ids, result_confidences, result_boxes

# format frame

def format_yolov5(frame):
    row, col, _ = frame.shape
    _max = max(col, row)
    result = np.zeros((_max, _max, 3), np.uint8)
    result[0:row, 0:col] = frame
    return result

# colors of bounding boxes

colors = [(255, 0, 0), (255, 165, 0), (255, 255, 0), (0, 255, 0), (0, 0, 255)]

is_cuda = len(sys.argv) > 1 and sys.argv[1] == "cuda"

net = load_model(is_cuda)

def detect_img(path, start):
    frame = load_img("test.jpg")

    # _, frame = img.read()
    inputImage = format_yolov5(frame)
    outs = detect(inputImage, net)
class_ids, confidences, boxes = wrap_detection(inputImage, outs[0])

end = time.time()
top = True #only save the top identified object
object1 = None
confidence1 = None
for (classid, confidence, box) in zip(class_ids, confidences, boxes):
    color = colors[int(classid) % len(colors)]
    print(classid, class_list[classid], confidence, color)
    if top==True:
        top = False
        object1 = class_list[classid]
        confidence1 = confidence

print("time elapsed:", end-start) #time to identify

for (classid, confidence, box) in zip(class_ids, confidences, boxes):
    color = colors[int(classid) % len(colors)]
    cv2.rectangle(frame, box, color, 2)
    cv2.rectangle(frame, (box[0], box[1] - 20), (box[0] + box[2], box[1]), color, -1)
    cv2.putText(frame, class_list[classid], (box[0], box[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 2)
    #print(classid, class_list[classid], confidence, color)
#cv2.imshow("output", frame)
#cv2.waitKey()
if (object1 is not None) and (confidence1 is not None):
    cv2.imwrite(" test.jpg", frame) #saves the bounding box image
    global path_to_img
    path_to_img = " test.jpg"
    global is_new_image
    is_new_image = True
    ...
    cv2.namedWindow("output", cv2.WINDOW_NORMAL)

# Show the Image in the Window
    cv2.imshow("output", frame)

# Resize the Window
    cv2.resizeWindow("output", 1920, 1080)

# Wait for <> miliseconds
    cv2.waitKey()
return object1, confidence1

# Setup Software
still WIP, since some setup should be in between functions :C, will leave this here for now

# [Button Setup]#
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BCM) # Use physical pin numbering
GPIO.setup(2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10 to be an input pin and set initial value to be pulled low (off)
GPIO.add_event_detect(2,GPIO.RISING,callback=button_callback) # Setup event on pin 10 rising edge

# Main Software

def main():
    Overview[0][0]="Object Detection"
    Overview[1][0]="Object Identified"
    Overview[2][0]="Object Located"
    Overview[3][0]="Object Grabbed"
    Overview[4][0]="Object Moving"
    Overview[5][0]="Object Dropped"
    Overview[6][0]="Cycle Complete"

    #forceThread = Thread(target=poll_force)
    #forceThread.start()

    while True:
        # ALL RED
        time.sleep(1)
        # object detection YELLOW
        set(0, 1)
        distance = poll_distance_until()
        # object detection GREEN
        set(0, 2)
        set(1, 1)
        start = time.time()
        print("taking picture")
```python
take_picture()
print("detecting picture...")
#object identified YELLOW
#set(1, 1)
object, confidence = detect_img(" test.jpg", start)
if confidence is not None:
    #object identified GREEN
    set(1, 2)
    global Guess
    Guess = object
    global Distance
    Distance = distance
    global Confidence
    Confidence = confidence
    #open gripper, close gripper, open gripper
    #object located GREEN
    set(2, 2)
    time.sleep(10)
    #object grabbed YELLOW
    set(3, 1)
    forceThread = Thread(target=poll_force)
    forceThread.start()
    open_steps()
    print("opened gripper")
    sleep(2)
    if object=="bodywash":
        newopen, newclose = grip_bodywash()
    elif object=="deodorant":
        newopen, newclose = grip_deodorant()
    elif object=="soap":
        newopen, newclose = grip_soap()
    elif object=="spray":
        newopen, newclose = grip_spray()
    elif object=="tea":
        newopen, newclose = grip_tea()
    #object grabbed GREEN, object moving GREEN
    print("closed gripper")
    set(3, 2)
    set(4, 2)
    time.sleep(5)
    #open gripper
    #object dropped YELLOW
    set(5,1)
    open_steps(newopen)
    print("opened gripper")
```
sleep(2)
global stop_forceThread
stop_forceThread = True
forceThread.join()
#object dropped GREEN, cycle complete YELLOW
set(5,2)
set(6,1)
time.sleep(2)
#close gripper
reset_close(newclose)
reset_close(200)
#cycle complete GREEN
set(6,2)
#don't want to remove picture, or else program will break looking for it
    #print("removing picture")
    #remove_picture(" test.jpg")
print("program end")
#global stop_forceThread
stop_forceThread = False
reset()

def main_image():
    start = time.time()
    print("taking picture")
take_picture()
    print("detecting picture...")
detect_img(" test.jpg", start)
    print("removing picture")
    remove_picture(" test.jpg")
    print("program end")

def main_grip():
    open_steps()
sleep(2)
    newopen, newclose = grip_soap()
    print(newopen)
    print(newclose)
sleep(5)
    open_steps(newopen)
sleep(2)
    reset_close(newclose)
sleep(2)
```python
mainThread = Thread(target=main)
mainThread.start()
GUI_Thread()
#main_grip()
```

Thread Software

```
<table>
<thead>
<tr>
<th>Part Number</th>
<th>Item</th>
<th>Supplier</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>101202</td>
<td>Camera Flex Cable</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/Adafruit-Flex-Cable-Raspberry-Camera/dp/B00XW2NCKS/ref=sr_1_3?crid=GS652C7UKH1&amp;keywords=RASPBERRY+PI+CAMERA+CABLE+2M&amp;qid=1651089041&amp;srefir=raspberry+pi+camera+cable+2m%2Caps%2C311&amp;sr=8-3#">https://www.amazon.com/Adafruit-Flex-Cable-Raspberry-Camera/dp/B00XW2NCKS/ref=sr_1_3?crid=GS652C7UKH1&amp;keywords=RASPBERRY+PI+CAMERA+CABLE+2M&amp;qid=1651089041&amp;srefir=raspberry+pi+camera+cable+2m%2Caps%2C311&amp;sr=8-3#</a></td>
</tr>
<tr>
<td>104303</td>
<td>Stepper Motor Driver 5pcs</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/HiLetgo-DRV8825-Stepper-RAMPS1-4-StepStick/dp/B01NCE3">https://www.amazon.com/HiLetgo-DRV8825-Stepper-RAMPS1-4-StepStick/dp/B01NCE3</a></td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>104304</td>
<td>Stepper Motor Wires</td>
<td><img src="https://www.amazon.com/RuiLing-Stepper-Motor-Cables-HX2-54/dp/B07QHTG82K/ref=pd_bxgy_img_1/142-3813915-72512637pd_rd_w=wpmav&amp;pf_rd_p=6b3eefed7-7b16-43e9-bc45-2e332cbf99da&amp;pf_rd_r=4FCV4W35Y455TRPGY453&amp;pf_rd_r=16cdfce3-4f9e-4762-835b-89400d9ec5b0&amp;pd_rd_wg=Jf5aP&amp;pd_rd_i=B07QHTG82K&amp;psc=1" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>104305</td>
<td>12V Power Supply</td>
<td><img src="https://www.amazon.com/Belker-Adapter-Supply-Charger-Electronics/dp/B08ZSDSXXL/ref=sr_1_4?crid=3TH8CVMHFCAX&amp;keywords=12+volt+power+supply+adapter&amp;qid=1646799150&amp;sprefix=12+volt+power+supply+adap%2Caps%2C360&amp;sr=8-4" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>105101</td>
<td>Pressure Sensors 2pcs</td>
<td>![Image](<a href="https://www.amazon.com/Bolsen-Tech-Pressure-Resistance-Resistor/dp/B098NZ6H2J/ref=asc_df_B07L6LV7G/?tag=&amp;&amp;linkCode=df0&amp;hvadid=343554957764&amp;hvpos=&amp;hvnetw=g&amp;hvrand=4963347596426843514&amp;hvqmt=3&amp;hvdev=c&amp;hvloc_interest=&amp;hv=10&amp;d">https://www.amazon.com/Bolsen-Tech-Pressure-Resistance-Resistor/dp/B098NZ6H2J/ref=asc_df_B07L6LV7G/?tag=&amp;&amp;linkCode=df0&amp;hvadid=343554957764&amp;hvpos=&amp;hvnetw=g&amp;hvrand=4963347596426843514&amp;hvqmt=3&amp;hvdev=c&amp;hvloc_interest=&amp;hv=10&amp;d</a> dog=4&amp;hvlocap=4&amp;hvarg=0&amp;hvtrack=1&amp;psc=1)</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Platform</td>
<td>URL</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>105102</td>
<td>Analog to Digital Converter</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/gp/product/B00NAY3RB2/ref=ox_sc_act_title_1?smid=A1VRRJFOL1VFP1B&amp;psc=1">https://www.amazon.com/gp/product/B00NAY3RB2/ref=ox_sc_act_title_1?smid=A1VRRJFOL1VFP1B&amp;psc=1</a></td>
</tr>
<tr>
<td>105201</td>
<td>Ultrasonic Sensor</td>
<td>Amazon</td>
<td>Amazon.com : WMYCONGCONG 10 PCS HC-SR04 Ultrasonic Distance Measuring Sensor Module for Arduino : Electronics</td>
</tr>
<tr>
<td>109100</td>
<td>RoboDk License</td>
<td>RoboDK</td>
<td><a href="https://robodk.com/pricing">https://robodk.com/pricing</a></td>
</tr>
<tr>
<td>N/A</td>
<td>Electrical Hardware Hit</td>
<td>Amazon</td>
<td><a href="https://www.vevor.com/belt-conveyor-c_10439/belt-conveyor-pvc-conveyor-belt47-x-7-8-inch-motorized-conveyor-w-guardrails-p_010756710227">https://www.vevor.com/belt-conveyor-c_10439/belt-conveyor-pvc-conveyor-belt47-x-7-8-inch-motorized-conveyor-w-guardrails-p_010756710227</a></td>
</tr>
<tr>
<td>104107</td>
<td>Gripping Material</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/dp/B08LQWR3B4/ref=cm_sw_em_r_mt_dp_5DQ8H3J9MY2Q9BC">https://www.amazon.com/dp/B08LQWR3B4/ref=cm_sw_em_r_mt_dp_5DQ8H3J9MY2Q9BC</a></td>
</tr>
</tbody>
</table>
FDM5X?_encoding=UTF8&psc=1

<table>
<thead>
<tr>
<th>104301</th>
<th>46 Ncm Nema Stepper Motors</th>
<th>Amazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>104301</td>
<td>56 Ncm Nema Stepper Motors</td>
<td>Amazon</td>
</tr>
<tr>
<td>104301</td>
<td>65 Ncm Nema Stepper Motors</td>
<td>Amazon</td>
</tr>
</tbody>
</table>

https://www.amazon.com/DROK-2-Phrase-Universal-Electric-Engraving/dp/B06XSYP24P/ref=sr_1_2_sspa?keywords=nema+17+stepper+motor&qid=1646799267&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzREVKUl1NklpNjYjImVuY3J5cHRlZElkPuEwOTk5ODA5MlhJWk1DUFUWFlyWSZlbmNyeXB0ZWRBZElkPUEwNjM5MTQzMTdUQVnZXRoyW1PWXwX2F0aGlhZ2VudG9zZWFyY2hpbmxpbmdsaWJsZS9mdWVsZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdEvxZ0NsaWNrPX RydWU=

https://www.amazon.com/STEPPERONLINE-Stepper-Bipolar-Connector-compatible/dp/B00PNEQKCO/ref=sr_1_4?crid=R6Z1B4HNB0I&keywords=nema+17+stepper+motor+high+torque&qid=1649288296&sprefix=nema+17stepper+motor%2Caps%2C153&sr=8-4#

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Provider</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>104305</td>
<td>M3 Screws</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/binifiMux-Phillips-Machine-Assortment-Stainless/dp/B07C96GV3H/ref=sr_1_20?crid=1B0FECCJ184VE&amp;keywords=m3+30mm+screw+assortment&amp;qid=1648500059&amp;sprefix=m3+30mm+screw+assortment%2Caps%2C170&amp;sr=8-17">Amazon Link</a></td>
</tr>
<tr>
<td>108101</td>
<td>Monitor</td>
<td>Best Buy</td>
<td><a href="https://www.bestbuy.com/site/samsung-43-class-7-series-led-4k-uhd-smart-tizen-tv/6401740.p?skuId=6401740">Best Buy Link</a></td>
</tr>
<tr>
<td>108102</td>
<td>Conveyor Belt</td>
<td>Vevor</td>
<td><a href="https://www.vevor.com/belt-conveyor-c_10439/belt-conveyor-pvc-conveyor-belt47-x-7-8-inch-motorized-conveyor-w-guardrails-p_010756710227">Vevor Link</a></td>
</tr>
<tr>
<td>108103</td>
<td>Assorted Zip Ties Pack</td>
<td>Amazon</td>
<td><a href="https://www.amazon.com/Assorted-ZipTie-Premium-Plastic-Outdoor/dp/B09247BXT1/ref=sr_1_2_sspa?keywords=zipties&amp;qid=1651090024&amp;sprefix=zip%2Caps%2C192&amp;sr=8-2-spons&amp;psc=1&amp;spLa=ZWNjcnlwdGVkUXVhbGlmaWVyPUEyT0hFTUc4UTJBUIASJmVuY3J5cHRlZmlnX3NjYW50YWdlNXJyMjU2M2FtZSk0ODA2OjE1MDAxQ0FJUTk1MDFRQjg2MjI3Q0VVQ0dKOEUxOEUyQ2Y1Q0U3M0dMNUE1Q0xReGQzQ1Q1T0hFQ0dKM0dK">Amazon Link</a></td>
</tr>
<tr>
<td>Code</td>
<td>Product Name</td>
<td>Supplier</td>
<td>URL</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>101101</td>
<td>Raspberry Pi HQ Camera</td>
<td>Vilros</td>
<td><a href="https://vilros.com/products/official-raspberry-pi-4-hq-camera?variant=31378811093086&amp;currency=USD&amp;utm_medium=product_sync&amp;utm_source=google&amp;utm_content=sag_organic&amp;utm_campaign=sag_organic&amp;gclid=CjwKCAjwtfqKBhBoEiwAZuesiM1wSE5LIUPIRPEPVZI5biqvARrR0Jht1LIPyavugiT0pPLCk4m1xoCNTwQA1vD_BwE">https://vilros.com/products/official-raspberry-pi-4-hq-camera?variant=31378811093086&amp;currency=USD&amp;utm_medium=product_sync&amp;utm_source=google&amp;utm_content=sag_organic&amp;utm_campaign=sag_organic&amp;gclid=CjwKCAjwtfqKBhBoEiwAZuesiM1wSE5LIUPIRPEPVZI5biqvARrR0Jht1LIPyavugiT0pPLCk4m1xoCNTwQA1vD_BwE</a></td>
</tr>
<tr>
<td>101201</td>
<td>6mm Camera Lens</td>
<td>Amazon</td>
<td><a href="https://vilros.com/products/official-raspberry-pi-4-6mm-wide-angle-lens?variant=31378836947038&amp;currency=USD&amp;utm_medium=product_sync&amp;utm_source=google&amp;utm_content=sag_organic&amp;utm_campaign=sag_organic&amp;gclid=Cj0KCQjwlOmLBhCHARIsAGlglq7kb6lAxXFqgePREPht3w7eHOVOQsJ5WDuxXMYM08833obhxyOK2ZSwaAnE5EALw_wcB">https://vilros.com/products/official-raspberry-pi-4-6mm-wide-angle-lens?variant=31378836947038&amp;currency=USD&amp;utm_medium=product_sync&amp;utm_source=google&amp;utm_content=sag_organic&amp;utm_campaign=sag_organic&amp;gclid=Cj0KCQjwlOmLBhCHARIsAGlglq7kb6lAxXFqgePREPht3w7eHOVOQsJ5WDuxXMYM08833obhxyOK2ZSwaAnE5EALw_wcB</a></td>
</tr>
</tbody>
</table>
Hardware Datasheet Links

Bolsen Force Sensor

Nema 17 46Ncm Stepper Motor

Nema 17 59Ncm Stepper Motor

Nema 17 65Ncm Stepper Motor
https://www.omc-stepperonline.com/download/17HS24-2104S.pdf

Raspberry Pi 4

Raspberry Pi 4 Camera Module

Ultrasonic Distance Sensor
https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
Maintenance

We made our gripper to be simple to use, easy to maintain and efficient in doing all tasks required. Our gripper is made of 14 individual parts and all these parts are connected using simple custom-made screws, screw holders and standardized screws.

Any part of these parts is easily removable and replaceable. We have 6 links parts that makes the movement of the gripper smooth, 2 gear links that transfer power from the main drive gear to the structure of the gripper, the palm that holds everything together and gives the gripper firm structure, the main gear that connects to the motor, the two fingers that hold our objects, the two fingertips parts that make sure we hold our objects firmly with the proper force and the handle to lift our gripper for demonstration and testing. While testing our gripper we realized that the most important parts that might face wear and tear are the motor and the fingertips. That’s why we decided to make these two parts the easiest to remove and change.

We have our drive gear that connects easily and removes easily to and from the shaft of the motor using a small standard M2 screw that holds it firmly but easy to remove if the motor fails. The fingertips parts are connected to the fingers using standard M3 screws to remove them since they are the only parts that touches the products and might face a lot of wear over time. The handle connects to the palm part using standard M3 screws that holds it firmly and makes sure it doesn’t drop or shake. All the other parts are connected using custom made screws that are easy and cheap to remove and re-attach by removing the screw holders with a sharp object and reattaching them to the end of the screws using glue.

Another important add-on is the material attached to the fingertips to add friction for holding our objects and those are attached using the glue that comes included with them and they are very easy and cheap to remove and change by peeling them off and attaching new ones to the outer side of the fingertips.
Extra Credit: Engineering Standards Used in the Adaptive Robot Gripper Project

ASME Y14.5: Geometric Dimensioning and Tolerancing

This engineering standard sets the symbols and datums for engineering drawings. It also covers tolerances of from, position, orientation, and profile. This sets the precedent for dimensioning requirements to allow product drawing and electrical drawing packages. Not only is this limited for setting the standards for drawings, but also verifications and requirements for manufactured parts.

This was used for the mechanical drawings of the gripper for this project. Reference Page 44

ASME Y14.100: Engineering Drawing Practices

This engineering standard is a guideline for the drawing practices of an engineer. This standard gives the criterion for symbols used, formatting, organization, and identification of an engineering drawing.

This was used for the mechanical drawings of the gripper for this project. Reference Page 44

IEEE 1016: Software Design Description

This Engineering Standard sets the requirement for the information content of the Software Design Documentation (SDD). It sets a standard organization for the content of the SDD along with the material that should be described within it.

This was used in the Software Design Document of this project. Reference Page 141