Barksdale®
CONTROL PRODUCTS

Final Report

Cobot ASV Failure Tracking System Implementation

ISE 495b Senior Design Project

Department of Industrial & Systems Engineering

Project Supervisor: *Sneha Lakshmi*

Coach: *Paul Lu*

By:

Iris Gordo, Devis Lai, Nayoung Oh, Ty Sawatyanon, Praneet Thirkateh

April 26, 2024

Table of Contents

Abstract **3**
Introduction **3**
Problem Statement **3**
Problem 3
Mission Objectives 4
Functional Requirements 4
Non-Functional Requirements 4
Design **4**
How the Failure Tracking System Works 4
Testing Results **7**
Future Recommendations **10**
Conclusion **12**
Appendix **12**
Our Google Drive Folder (video tutorials, reports, etc.) 12
Updated User Guide Manual 2023-2024 (also found below) 12
Customer Approval **19**

Abstract

Barksdale is a subsidiary of Crane Co. located in Vernon, California which manufactures high-performance fluid control and measurement equipment such as valves, pressure/hydrogen, IIoT, Air Suspension Valves (ASVs), and more. Under the guidance of the Value Stream Manager at Barksdale, Sneha Lakshmi, our team was asked to improve the testing process for their ASVs by interfacing with Cobots (Computer-controlled robotic devices) which conduct the tests. Building upon the groundwork accomplished by the previous student group, we have been assigned to implement and standardize a failure tracking system as well as update the user manual to be more comprehensive to assist employees and future students in implementing the new system and navigating issues. Due to time constraints and the lack of specifications in the previous group's manual, we were not able to achieve adequate results when running the failure tracking system. However, we have curated an enhanced user manual with detailed instructions, videos, and illustrations to continue this project further and ensure that future groups can focus on collecting data from the tracking system, overall alleviating the challenges we faced during setup.

Introduction

In this project, our project supervisor Sneha Lakshmi asked us to focus on the quality tests that Air Suspension Valves (ASVs) go through. These valves are evaluated through a Cobot which conducts three tests: dump test, rotor test, and deadband test. For reference, these tests were explained to us by ASV Supervisor Bryan Hernandez.

Problem Statement

Problem

Barksdale is trying to automate their lines for the transportation department. The current quality testing process can track the number of failures but cannot record the failure modes for each specific valve. Moreover, they collect false positives when assembly line workers physically hit a failed valve against a table, pass the quality test, and call it "rework" and is not recorded as a failed valve. As a result, their failure rate inaccurately lowers. Being able to distinguish failure modes will help Barksdale identify root causes to why each valve fails quality tests.

The previous student group that worked on this project created a failure tracking system using Python to track what failure occurred with each valve that passes through the Cobot. However, they were not able to validate their tracking system using a large enough sample size.

With that, the following is a list of our objectives.

Mission Objectives

1. **Validation** – Ensure the failure tracking system provides accurate results (number of failures, types of failures, etc.).
2. **Repeatability** – Standardize the implementation of the failure tracking system and apply to Lines 2, 3 and 4.
3. **Simplicity** – Ensure the data summarized and presented on PowerBI is easy to understand for all employees.

Functional Requirements

1. Historical database containing reasons for failure, failure rate, etc. for L2, 3, and 4
2. Summarized data on PowerBI that is simple and accessible to all.
3. Reduced Cobot failure rate*
4. Reduced number of units scrapped*
5. Reduced operational costs*

*Requirements with the asterisk are to be listed as specific quantitative baselines provided by the Project Supervisor. Overall, the main goal of the project should be to reduce the failure rate.

Non-Functional Requirements

1. Cobot
2. Computer
3. Software - Excel, PowerBI, Tracking system (Python code)
4. Barcode Scanner/Stickers
5. User Guide for Tracking System Setup (2022-2023 USC Student Group)

Design

How the Failure Tracking System Works

The following describes the tracking system after following the Cobot and computer setup instructions in the *User Guide Manual* below at the Appendix section.

Once the Cobot and computer are properly set up and the program is running, the user may pass a valve through the Cobot. When collecting data, it is best to use failed valves since that is the primary focus of the tracking system. When the Cobot identifies the valve as a failure, stick a barcode to the valve and scan it with the barcode scanner. Afterwards, the program should save the valve's barcode ID and the failure type to a csv file. This was as much as we understood about the program. More details about how the program works can be found in the previous group's final report which is found in the Google Drive linked in the Appendix.

The following process maps illustrate how we understood the process of the quality test before and after implementing the tracking system.

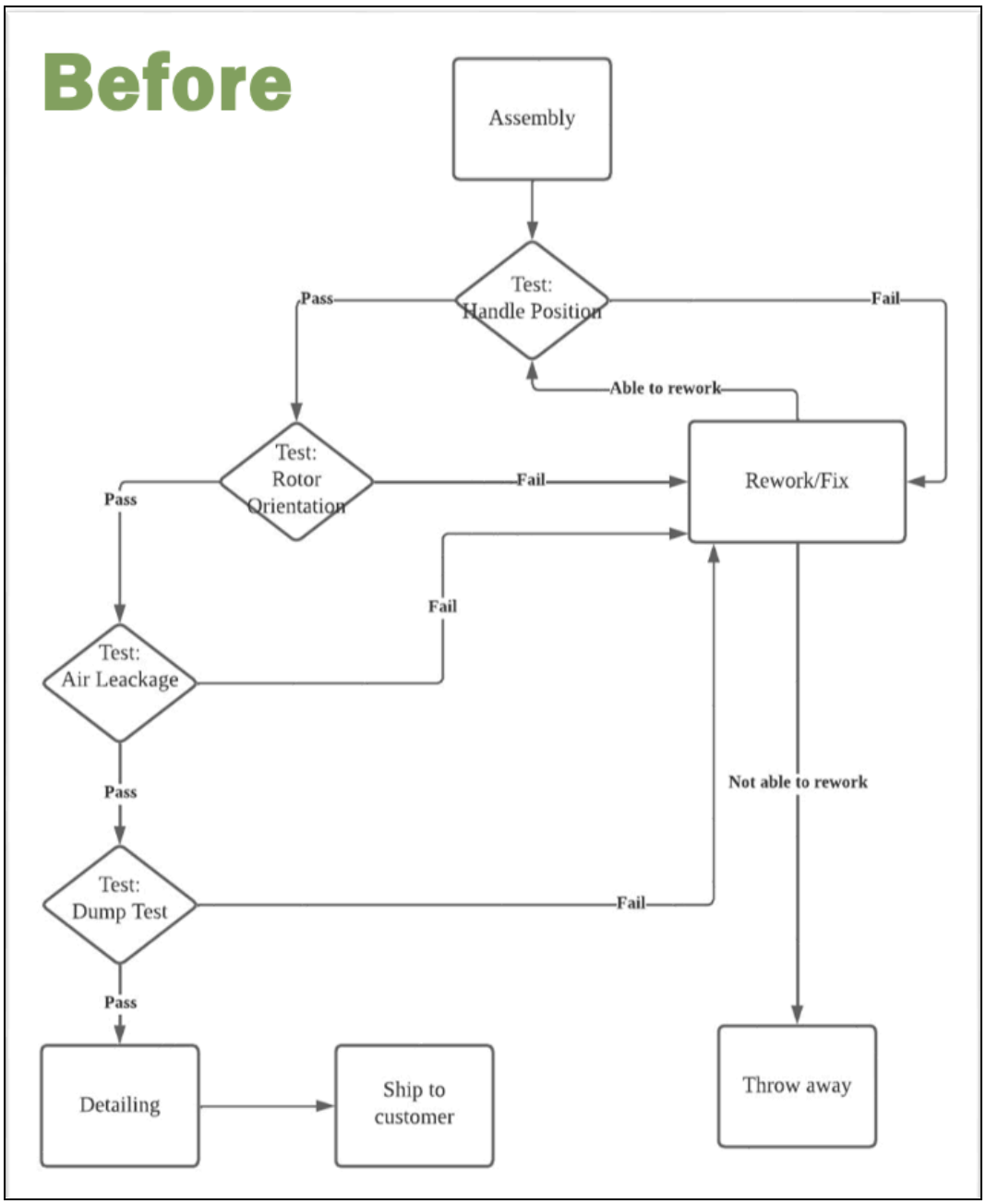


Figure 1: Process Map Before Tracking System Implementation

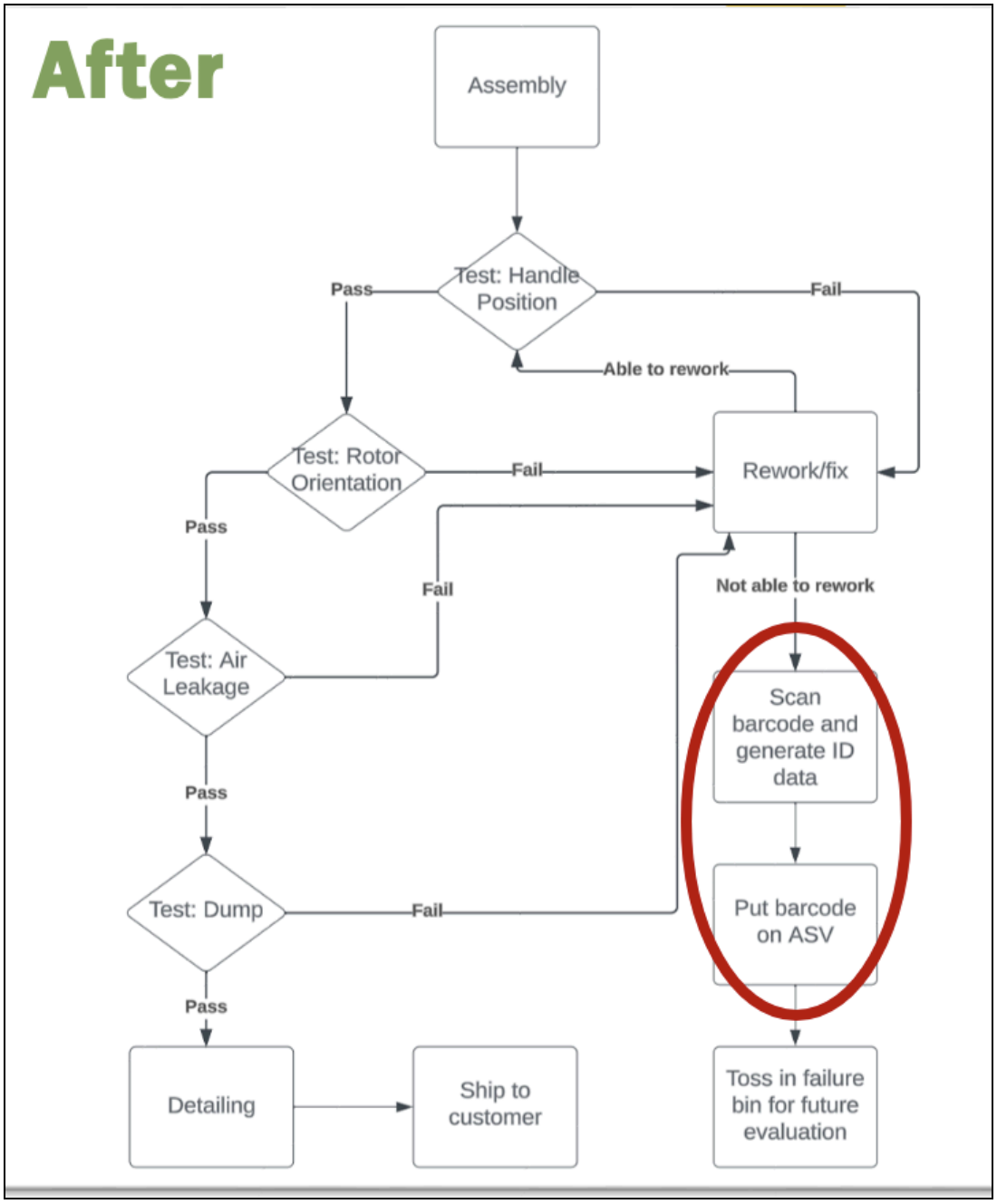


Figure 2: Process Map After Tracking System Implementation

Testing Results

We were able to complete all initial steps for the connection setup between the Cobot and computer and proceeded to testing the tracking system. However, due to time constraints and the vagueness of the previous team's user manual, our team faced unknown errors and was not able to fix them on time. Some of these errors, we assumed, were technical errors due to something wrong in the connection that we established, such as a mistake in one of the testing code files like record_test.py (which is still the most likely culprit), or on the cobot's network settings which can only be accessed through code but we would need to understand this more through Universal Robotics' online forums.

There were several troubleshooting attempts in modifying the Cobot code and Python code For example, in the Python program, we changed the frequency (as instructed in the original manual) as it stated that the frequency changes how quickly data is collected and increasing it may reduce errors. However, we still received the same errors as shown below. We also attempted printing the different variables that were assigned in the Python program, however, the results were unclear (Figure 4) and did not eliminate the errors.

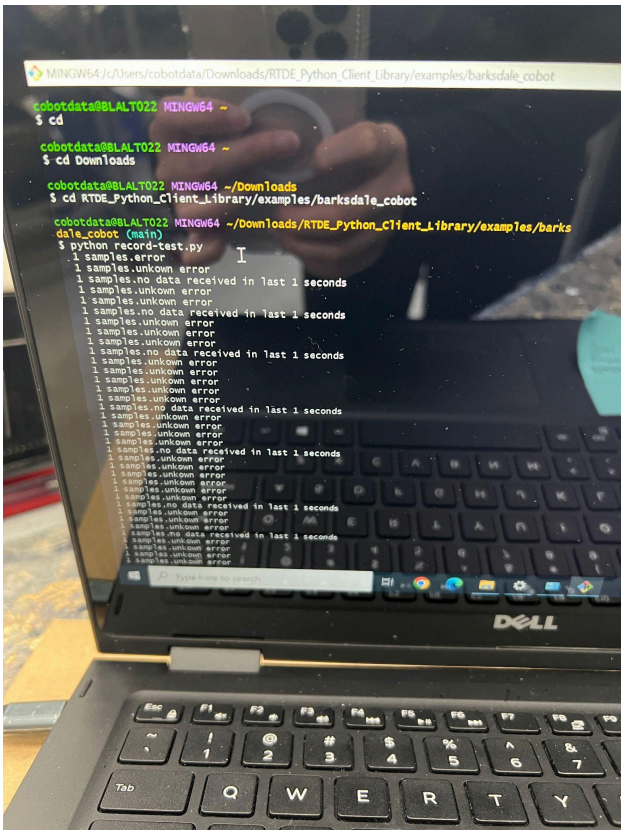


Figure 3: Error #1 - Testing Different Frequencies

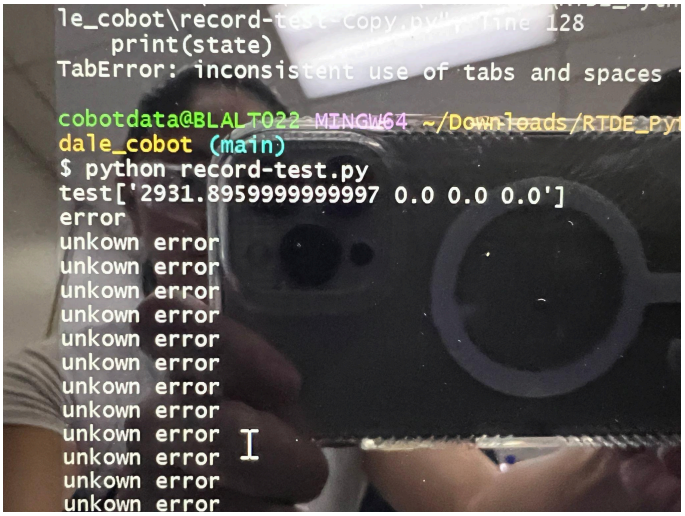


Figure 4: Error #2 - Executed *print("test"+str(data_list))* on Line 151

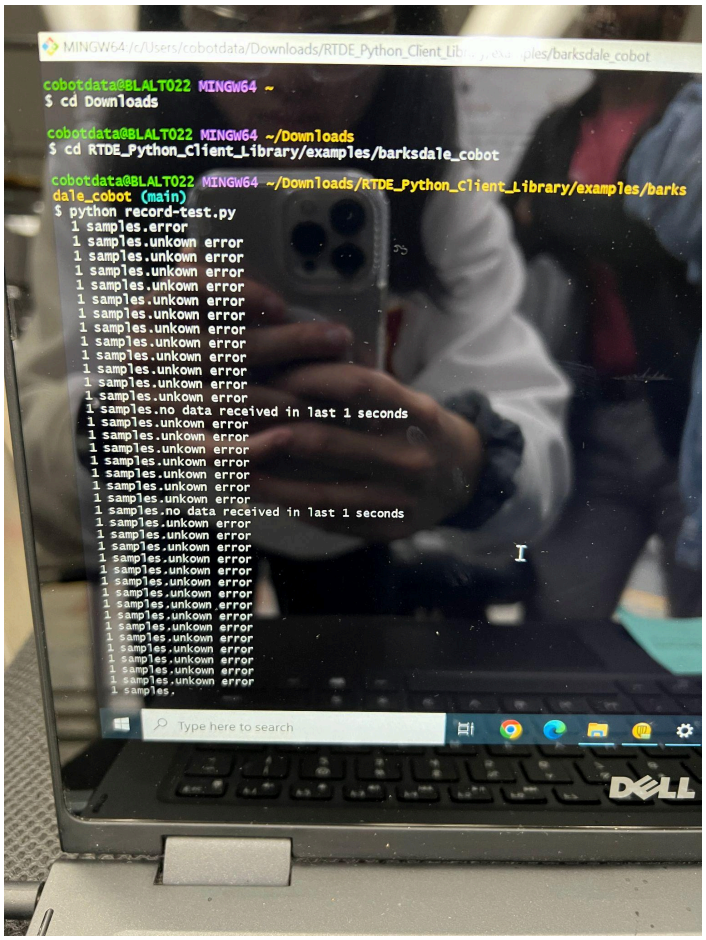


Figure 5: Error #3 - Testing Different Frequencies

Figure 6: Error #4 - Testing Different Frequencies

As we can see above, there are a plethora of errors that we encounter once the connection is actually set up, but troubleshooting using Eduardo's help and using any official Universal Robotics Forums should help work around some of these errors. It is paramount that anybody working on this project is able to complete the full hardware and software set up and complete the connection between the Cobot in a very timely manner (ideally within 1.5 months of starting the project) so that they can troubleshoot these unknown errors in the final stages of testing the connection. It is also paramount that you try to schedule as much time as you can to get cobot availability as they are usually being used for work and are unavailable for our testing except for specific times during the weekdays.

Once the connection is established and you are comfortable getting consistent quality testing results for the ASV's, you must focus on creating a procedure for actually implementing this failure tracking system into one of Barksdale's manufacturing lines with an automated quality test (cobot). This will require you to set up specific operational procedures and suggest training

methods to Barksdale for them to implement it at a very low cost. This is the next phase of the project.

Future Recommendations

The root cause for why we were unable to obtain adequate results and troubleshoot the program is because of the vague instructions provided by the previous team which lead to nearly two semesters of solely attempting to prepare the Cobot and computer interfaces for data collection. To prevent this issue for future teams, we have curated our own set of instructions which is more detailed and include illustrations that should allow anyone to set up the Cobot and computer connections as well as run the program in less than 10 minutes. We highly recommend using this updated manual for preparation to focus on the data collection phase. All steps will help future teams achieve the following:

1. Prepare the computer for data collection
2. Connect the computer with the Cobot externally
3. Edit the Cobot program for quality tests
4. Connect barcode scanner to computer
5. Run the Python program

When ready to collect and analyze data, refer to the previous group's instructions as we were unable to recreate instructions for those phases. **Please note that while our instructions are more detailed, there may be some slight inaccuracies since the Python program was outputting errors and one of the members of the previous group James Huang stated that running the program should not have any issues.**

As we observed in the previous section, there are a plethora of very similar errors that we encounter once the connection is actually set up, but troubleshooting using Eduardo's help and using any official Universal Robotics Forums should help work around some of these errors. It is paramount that anybody working on this project is able to complete the full hardware and software set up and complete the connection between the cobot in a very timely manner (ideally within 2 months of starting the project) so they can troubleshoot these unknown errors in the final stages of testing the connection. It is also paramount that you try to schedule as much time as you can to get cobot availability as they are usually being used for work and are unavailable for our testing except for specific times during the weekdays.

Once the connection is established and you are comfortable getting consistent quality testing results for the ASV's, you must focus on creating a procedure for actually implementing this failure tracking system into one of Barksdale's manufacturing lines with an automated quality test (cobot). This will require you to set up specific operational procedures and suggest training methods to Barksdale for them to implement it at a very low cost. This is the next phase of the project.

If the program still outputs errors even after troubleshooting, we ***highly recommend*** creating another simple process that operators can follow to track ASV failures. It can be as simple as creating a barcode per test (rotor, deadband, dump) and having the operators record the failures on a spreadsheet or even on paper if necessary. The main goal of the project was to implement the failure tracking system and standardize this process so that it may be implemented into multiple manufacturing lines, so it is paramount that we come up with a process that would work easily and which the operators would have no problem using daily. Any system that gets the job done should be prioritized, even if it isn't very complex (like software, which we have pursued). We failed to meet this objective since the challenges of the program distracted us. Although it would be efficient to automate, it's more important to accomplish this goal and to be open-minded when doing so. In fact, after presenting our final report to our project supervisor Sneha, she stated that she would have accepted a more manual approach as long as it accomplished the project goals. **Overall, ensure to always have the project goal in mind.**

Finally, we visited Barksdale at least twice a week. We recommend doing the same as it may take some time understanding the processes for the quality test, getting familiar with employees and their roles, and understanding the project requirements and parameters overall. Moreover, a Cobot must be available to run tests, meaning that it is not being used during production hours. Therefore, expect difficulty when scheduling visits with Cobot availability because although Barksdale has a production schedule, the schedule changes frequently and cannot guarantee availability during your free time. However, a Cobot will always be available daily from 2:30-3:15pm as they conduct changeovers during this time. This may change, but we recommend staying in touch with Eduardo, an Engineer from Barksdale, regarding Cobot availability. The time may be tight, however, it was enough opportunity for us to understand and complete all steps for setup. To prevent team scheduling conflicts, we recommend determining a time that at least 2-3 students can visit and work on the project. Contact Eduardo at least one day prior to your visit so that he is aware and can let you know if Cobots won't be available.

Barksdale Contacts:

- Sneha Lakshmi - Project Supervisor
    - She initiated the project and can tell you what the objectives are. You will be presenting your project milestones to her throughout the school year.
- Eduardo - Engineer
    - He was our main point-of-contact throughout the project since he was familiar with the Cobot's functions and components. We recommend coming to him for technical questions first.
- Ronald - Engineer
    - He is also very familiar with how the Cobot operates and he can be contacted as a second resource for more questions.

USC Senior Design Group:
You may contact us for additional inquiries. James Huang is also a relevant contact as he created the Python program and participated the most in the project the year before us. His contact information is below.

- Iris Gordo | 510-331-2239 | iris.rgordo@gmail.com
- Praneet Thirkateh | 630-379-4013 | thirkate@usc.edu
- Devis Lai | devislai@usc.edu
- Nayoung Oh | nayoungo@usc.edu
- Ty Sawatyanon | sawatyan@usc.edu
- James Huang | hjames034@gmail.com | jhhuang@usc.edu

Conclusion

Despite facing challenges such as time constraints, limited cobot availability, and grappling with manual comprehension and connection debugging, we unfortunately couldn't finalize testing on line 2 or implement the system on lines 3 and 4. However, we took the opportunity to refine the manual, ensuring it is more comprehensive and up-to-date. The biggest lesson learned is that we could have simplified the process, yet we made it more complicated by focusing too much on how to make the program work, resulting in not meeting the project goals. If we had the opportunity to spend more time on this project, we would have pursued building a simpler system that achieved our objective to track ASV failures, even if it meant abandoning the complicated software we had worked on until then. Barksdale has given us the greatest opportunity to work in the real world and learn life-lessons. We extend our sincere gratitude and joy to Barksdale for hosting the project, and we appreciate their support to help us grow as engineers, students, and individuals.

Appendix

Link 1: Our Google Drive Folder (video tutorials, reports, etc.)

Link 2: Updated User Guide Manual 2023-2024 (also found below)

User Guide Manual 2023-2024:

Setup

1. **Gather the necessary materials. (Eduardo knows what you need)**
    a. 2 ethernet cables
    b. 1 network switch
    c. 1 computer (used by previous group)
    d. 1 adapter
    e. 1 USB-A keyboard with extra side number pad
    f. 1 Cobot (any that is available)
        Note: we used cobot TUCO-6601 most of the time since this is what the previous group used, but we also used TUCO-6604 since they are generally the same with slightly different programs. All other cobots were not tested
2. **Prepare Computer to Interface with Cobot**
    a. On the computer, navigate to the *barksdale_cobot* folder.
    b. Open the r*ecord_configuration.xml* file and ensure that registers 2, 3, and 4 are not commented as shown below:

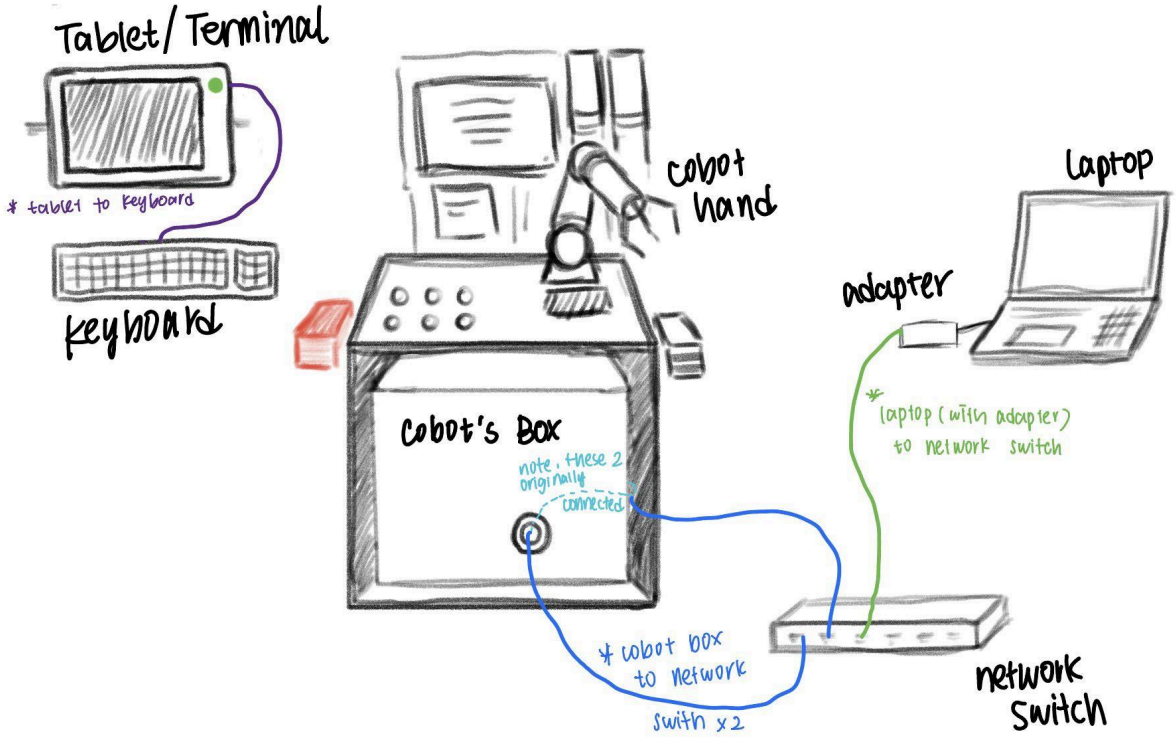
```
<?xml version="1.0"?>
<rtde_config>
    <recipe key="out">
        <field name="timestamp" type="DOUBLE"/>
    <!--
        <field name="output_double_register_1" type="DOUBLE"/>
    -->
        <field name="output_double_register_2" type="DOUBLE"/>
        <field name="output_double_register_3" type="DOUBLE"/>
        <field name="output_double_register_4" type="DOUBLE"/>
    <!--
        <field name="output_double_register_5" type="DOUBLE"/>
    -->
```

        i. Note: A “comment” in the code will not run during the test. To make a comment, simply type “<!--” before the code you want to comment and “-->” after. For example, the comments in the picture are gray while the colored text is not commented. Everything else in the file should remain as it was. This step may be completed already since we saved all files.
        ii. The purpose of this step is to save colleged data to registers 2, 3, and 4. Each register is assigned a type of test.

3. **Connect Computer to Cobot Externally**

   Step 3 Video Tutorial

   Refer to the following figure.



   a. Connect the adapter to the computer (Note: the adapter is a bit loose and might cause the computer/laptop to lose connection with the Cobot. Make sure that the lights on the adapter are green which indicates that it is connected)
   b. Disconnect the ethernet cable under the Cobot's box and connect it to the network switch. The cable is usually either on the bottom or on the side of the box as illustrated in the picture. The cable should be blue.
   c. 1st Ethernet cable from **Step 1** - Connect one end to the previous step's port (Cobot box) and connect the other end to the network switch.
   d. 2nd Ethernet cable from **Step 1** - Connect one end to network switch and the other end to the computer's adapter.
   e. Connect the keyboard to the Cobot's tablet as shown on the figure above.

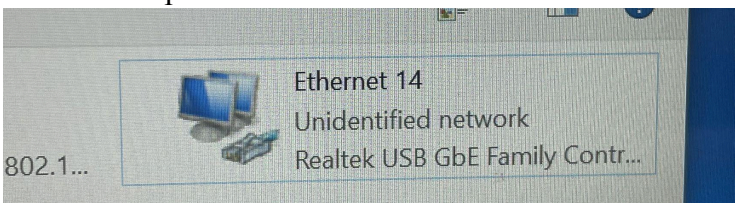
4. **Connect Computer to Cobot via Ethernet**

   Step 4 Video tutorial

   a. On the computer, navigate to *Settings* application and follow these steps:
      i. *Network & Internet* → *Change Adapter Options* → *Right click on "Ethernet Connection"* → *Properties* → Enter security login provided by Eduardo or another Engineer/IT staff. This step may not be necessary. Sometimes, the ethernet is connected perfectly and you can confirm that with the picture on **Step 4a iii**.
      ii. Ensure that the IP address on the computer is different from the Cobot's IP

address by changing the last digit to an arbitrary number. For example, if the IP address on the Cobot is 100.100.100.7, change the IP address of the computer to 100.100.100.5. You can change this under *Properties*. Ask Eduardo for help if you're unable to fulfill this step.

iii. Refer to this photo make sure that ethernet is connected:

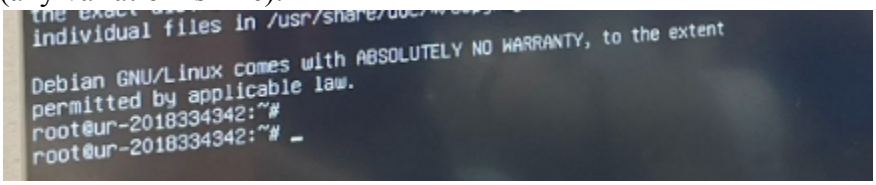


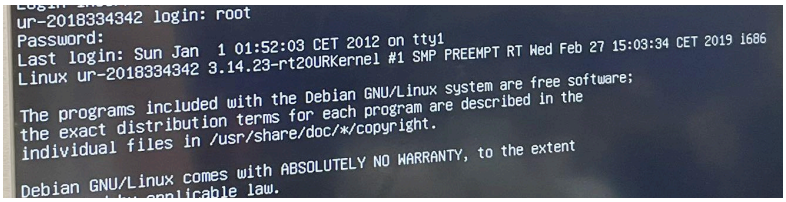
If the ethernet is not connected, you will see a red "X" instead of an ethernet icon on the bottom left corner of the network icon.

5. Setting up Cobot-Computer Connection

Step 5 Video tutorial

a. Turn on the Cobot (top right corner of tablet).
b. On the keyboard, do the following:
   i. Hit **CTRL + ALT + F1** (at the same time). This should display a terminal/cmd-like window on the tablet.
   ii. Enter the username/password (press enter after each step).
      1. *User: root*
      2. *Password: easybot*
         a. Note: the password won't be displayed for security reasons
         b. If the password is invalid, ask relevant individuals such as Eduardo or Bryan Hernandez to find it.
      3. This step is complete if the screen appears similar to this figure (any variation is fine):





   iii. Run the following code. Since the keyboard is not properly configured, you will need to utilize the ASCII code equivalent to some characters. We used the source below. We have also provided the original code and the ASCII code version below:
   https://www.irongeek.com/alt-numpad-ascii-key-combos-and-chart.html
      1. Original code (displayed on terminal):
         a. : echo 1 > /proc/sys/net/ipv4/ip_forward
         b. iptables -A OUTPUT -p tcp --sport 30004 -j ACCEPT
      2. ASCII code (typed on keyboard)

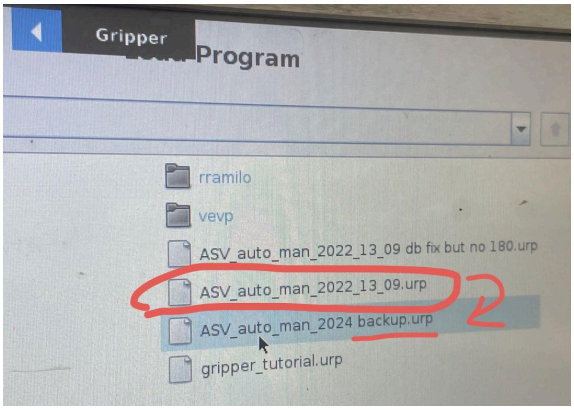
a. [Shift]+[period] [space] echo [space] 1 [space] [alt]+62 [space] [alt]+47 proc [alt]+47 sys [alt]+47 net [alt]+47 ipv4 [alt]+47 ip [alt]+95, forward
b. Have another member that you validate that you entered the correct sequence in part A. Then, hit enter and continue to part c.
c. iptables [space] [alt]+45 A [space] OUTPUT [space], [alt]+45 p [space] tcp [space] [alt]+45 [alt]+45 sport [space] 30004 [space] [alt]+45 j [space] ACCEPT
d. Notes:
    i. [space] = one click on the space bar (don't actually type the brackets and the word space, brackets are indications of keys)
    ii. The numbers after alt MUST be typed in the right number pad on the keyboard (not the top row of numbers)
    iii. Don't type in the plus signs. For example, [alt]+45 means you hit [alt] key and hold it until you hit 45
    iv. Any plus sign combinations should be typed at the same time
    v. The terminal should return something similar to this: *root@ur-2018334342:~#*. Minor differences are fine as they are caused by using different cobot systems.
    vi. If the same prompt appears after typing the first line, then go to the next step. If the same prompt appears after you typed in the second line, all is done and you can exit the terminal
    vii. If anything is typed wrong, there will be an error message but you can try again starting from part 2a.

3. Exit the terminal by typing **CTRL+ALT+F7** (at the same time)

6. Adding Code to the Cobot

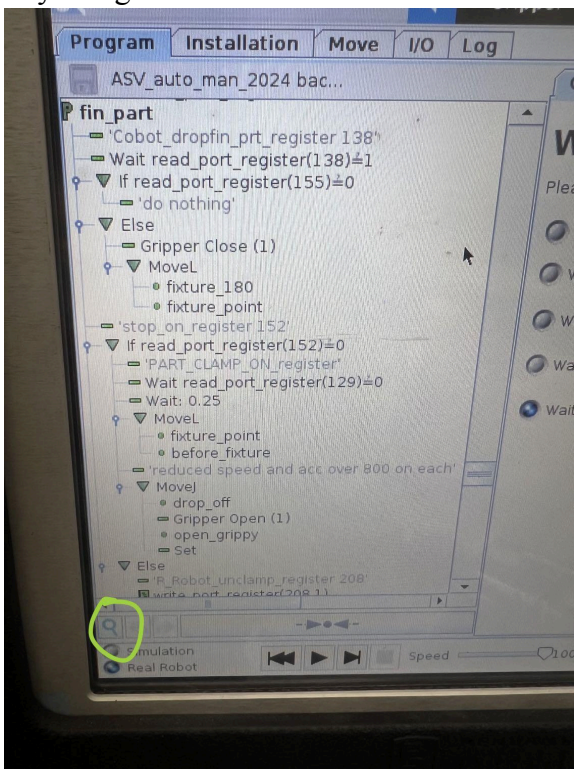
Step 6 Video tutorial (adding code to the Cobot only)
How to add script code to Cobot
How to delete script code on Cobot

a. On the Cobot tablet, do the following:
    i. *Program Robot → Load Program → [ASV Program Name]*. Note: Use this ASV file circled in red and make a backup of this. Cobot TUCO-6601 has the backup we used and you can edit as necessary.



ii. To make a backup, click on *File* → *Save As*, and make a copy of the ASV program. This ensures that should anything go wrong while adding lines of code to the program, we can leave the original program intact.

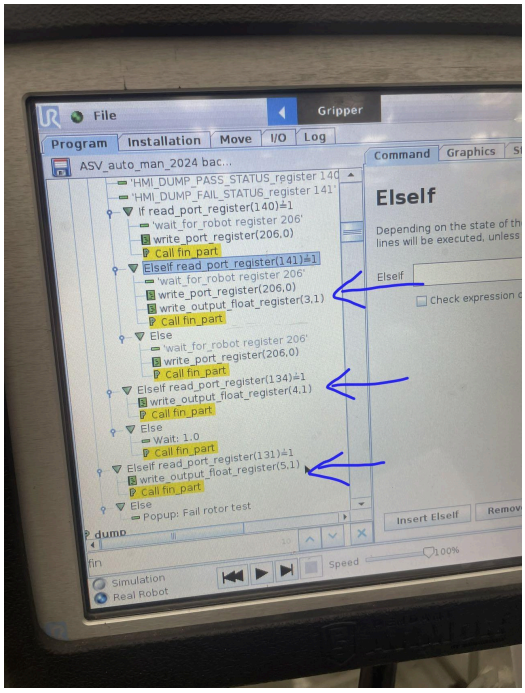
b. Locate areas of the code that have the syntax *Call fin_part* (or similar syntax like fin_partUSB on Cobot 6604, they should all represent the same command). You can accomplish this by using the search bar on the bottom left of the screen.



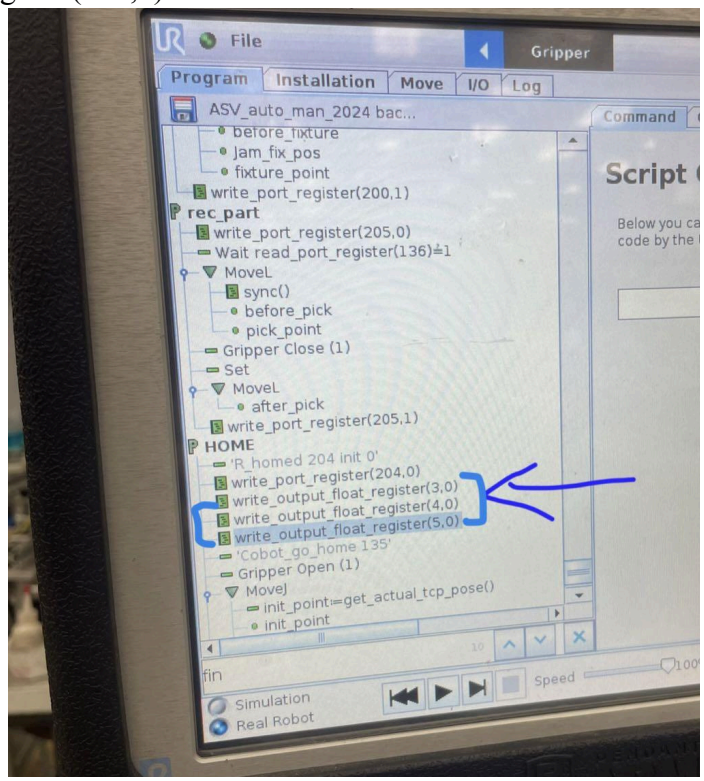
c. There should be 3 areas where this line of code exists in the program. For each instance, insert Script Code. In the following order, add:

i. write_output_float_register(2,1)
ii. write_output_float_register(3,1)
iii. write_output_float_register(4,1)

Here's what we did on Cobot 6601 (add the 3 lines to where the arrow points which should be in the Robot Program function):



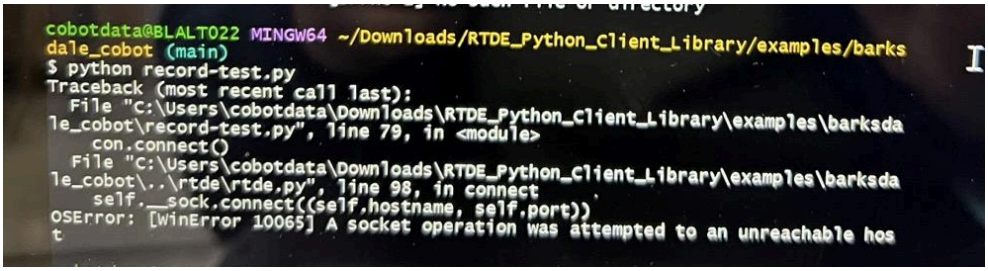
d. Locate the HOME function at the end of the script and add another Script Code write_port_register(204,0)



7. **Connect Scanner to Computer**
   a. Follow the same instructions from the previous group. We had no issues with these steps.

Run

8. **Run the Program**
   Step 8 Video tutorial
   a. Open Github bash
   b. Before running the program, you must first locate the folder the python file is in. You can do so by typing the following:
      i. cd Downloads/RTDE_Python_Client_Library/barksdale_cobot
   c. Run the program by typing the following:
      i. python record-test.py
      ii. If the terminal reports an error similar to "socket operation unreachable" like this:



      it either means the computer or the cobot system is disconnected and the systems are searching for connection. Check both the network switch and the adapter and make sure the lights are green (they are connected), you can also try checking step 4.a.iii. It's usually the adapter being disconnected.

Customer Approval

Name	**Signature**
Sneha Lakshmi