STUDENT SUMMER INTERNSHIP TECHNICAL REPORT

Integration and Automation of Pipe Crawler and Reel System for Lateral Gamma Scanner

DOE-FIU SCIENCE & TECHNOLOGY
WORKFORCE DEVELOPMENT PROGRAM

Date submitted:

December 16, 2022

Principal Investigators:

Josue Estrada (DOE Fellow Student)
Florida International University

Douglas Reid Ph.D. (Mentor)
Washington River Protection Solutions

Ravi Gudavalli Ph.D. (Program Manager)
Florida International University

Leonel Lagos Ph.D., PMP® (Program Director)
Florida International University

Submitted to:

U.S. Department of Energy
Office of Environmental Management
Under Cooperative Agreement # DE-EM0005213

Applied Research Center
FLORIDA INTERNATIONAL UNIVERSITY

EXECUTIVE SUMMARY

This research work has been supported by the DOE-FIU Science & Technology Workforce Development Initiative, an innovative program developed by the U.S. Department of Energy's Office of Environmental Management (DOE-EM) and Florida International University's Applied Research Center (FIU-ARC). During the summer of 2022, a DOE Fellow intern, Josue Estrada, spent 10 weeks doing a summer internship at Washington River Protection Solutions at the Hanford Site under the supervision and guidance of Dr. Douglas Reid. The intern's project was initiated on June 06, 2022, and continued through August 12, 2022, with the objective of furthering the development of an automated inspection tool for detecting gamma radiation leaks underneath the tanks at the 241-A and SX tank farms.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The goal of this project was to create a robotic inspection system that can routinely and autonomously scan for an increase in radiation under the tank farms at the Hanford Site. The source of this radiation would be from waste leaks during retrieval operations. The waste leak would indicate damage to the tanks and must be detected as soon as possible. However, these tanks are buried underground, leaving no access points around or underneath them except for their 3-inch diameter horizontal lateral pipes, as shown in Figure 1 in plan and Figure 2 in elevation.

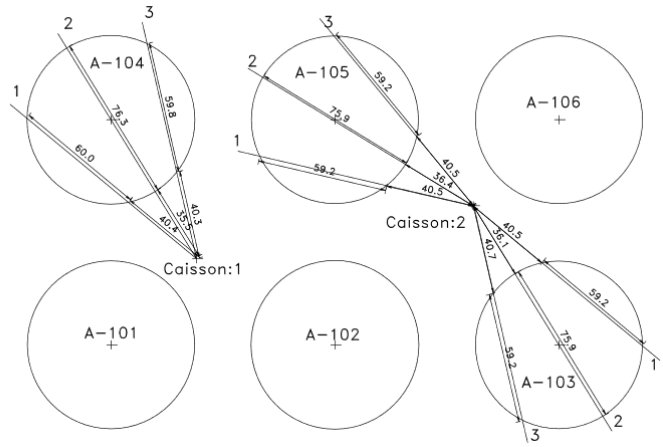

Figure 1. Tank farm diagram showing lateral channels

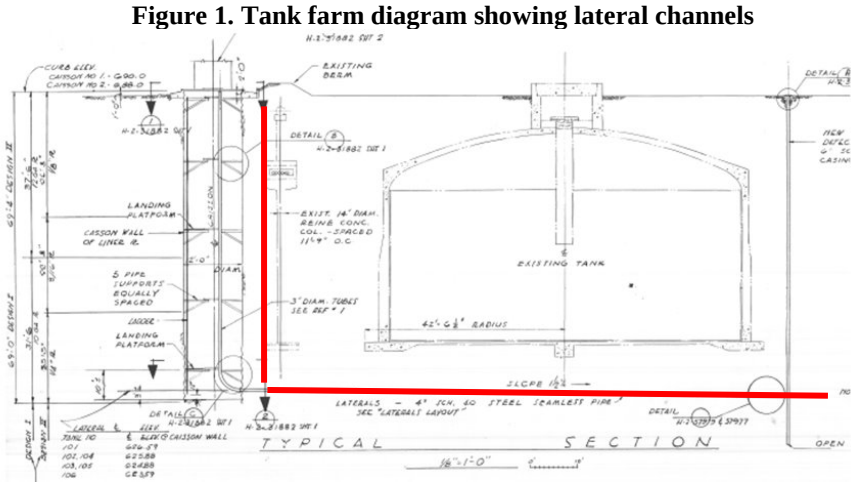

Figure 2. Individual Tank Schematic

Figure 2 shows that the lateral pipes are situated 10 feet below the tanks and span the entire diameter of each tank. To access these pipes from the surface, a caisson was installed during the farm's construction. While these lateral pipes have been utilized in the past to measure radiation, the process required a large team of operators to conduct the measurements. In order to address this issue, the Robotics Laboratory at FIU's Applied Research Center is collaborating with Washington River Protection Solutions to develop an inspection tool that can utilize these lateral pipes.

The proposed inspection tool would be able to transport gamma radiation sensors along the lateral pipe by utilizing a robotic system that integrates a peristaltic-motion pneumatic pipe crawler and

a mechanized reel into a single, automated delivery system. The pneumatic-based locomotion system is more dependable and adaptable to a variety of friction and inner-pipe surface conditions than a wheeled or track-based solution. Additionally, the mechanized reel can retrieve the crawler and sensors following scanning. By implementing an automated system, multiple consecutive scans can be conducted daily without the need for active and costly operator involvement. The team expects to create a resilient inspection system that can carry out multiple inspections daily, record the results of scans across the tank diameter, and retrieve itself in the event of a malfunction.

Initially, the focus of this internship was on developing the pipe crawler and reel components, along with the objective of integrating them into an automated system. As a result, this report will concentrate on the integration and management of the lateral gamma scanner system (LGS).

2. RESEARCH DESCRIPTION

2.1 System components

Prior to this internship, the Robotics Lab at FIU's Applied Research center designed and built a platform for transporting cylindrical gamma sensors through the lateral channels. Figure 3 depicts a model of the delivery section of the crawler that makes up the platform. The pipe crawler comprises several sections, and individual modules within each section are linked together with connections that are rigid along the axis. This enables the delivery crawler section to propel all the sections in front of it forward using peristaltic motion.

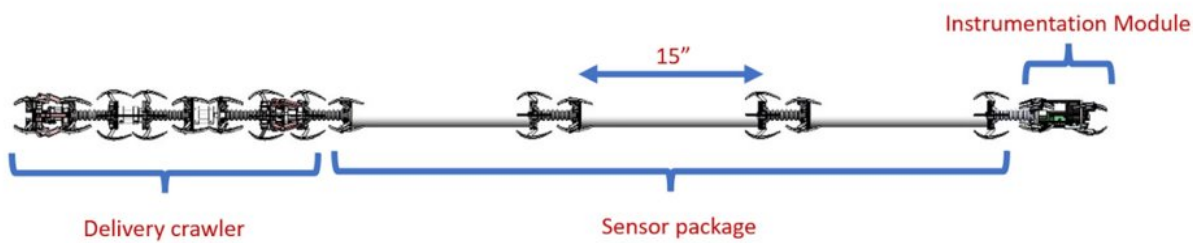

Figure 3. Pipe crawler sections (the rightmost module is at the crawler's front

Peristaltic motion is executed by combining gripper modules that grip or release the inner wall of the pipe as needed, and extender modules that elongate the crawler. The extender modules are located at the front and back of the delivery section and in between them. To begin a cycle of forward crawl, the back gripper is first engaged to secure the position of the crawler inside the pipe. Then, the extenders are activated, elongating the crawler forward and pushing the frontal sections (displayed upfront in Figure 4). The front gripper is then engaged, and the back gripper is disengaged to fix the front position of the crawler. The extenders are then compressed to complete one cycle of the forward crawl. This cycle is illustrated in Figure 5.

Figure 4. Peristaltic Motion Pipe Crawler

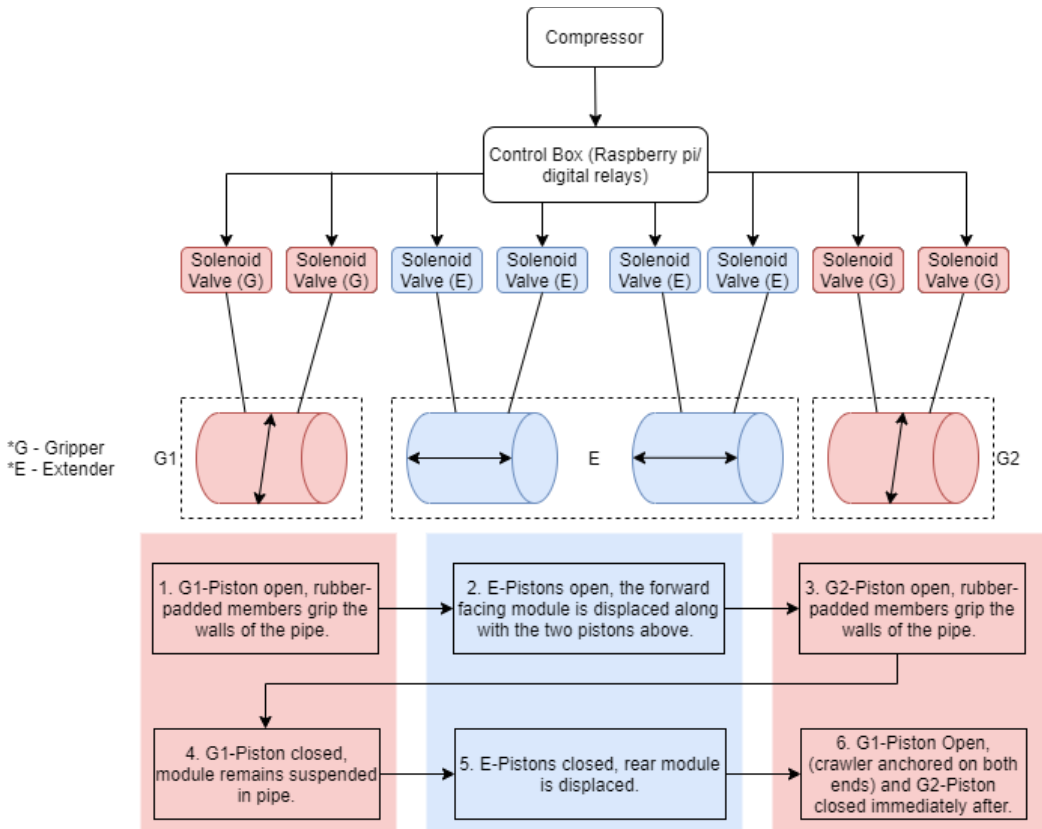

Figure 5. Diagram of peristaltic motion components and operation

It is important to note that the crawler's module configuration is symmetrical, allowing it to move in both forward and backward directions as required. The pistons can be actuated individually, enabling the system to completely disengage the grippers and retrieve the crawler using the mechanized reel, which is the current plan for robot recovery.

At the front of the crawler's section, there is an instrumentation and sensor module as shown in Figure 6. This module contains a Raspberry Pi Zero embedded computer, which assists in navigation by probing the necessary sensors when required to interact with the pipe environment.

Figure 6. Lateral Gamma Scanner Front Module

The first sensor is a wide-angle-lens camera that serves a dual purpose of recording the inspection video and providing visual odometry. Additionally, the module incorporates a 1D LiDAR that acts as a rangefinder by measuring the time of flight of laser pulses in the forward direction to determine the distance to the end of the pipe or the bottom elbow during the vertical section. Furthermore, the module utilizes an inertial measurement unit consisting of gyroscopes and magnetometers to calculate the relative inclination angles with respect to earth, linear accelerations, and angular velocities, which can be used to estimate the crawler's position. Lastly, an environmental sensor is available to read values of temperature, pressure, humidity, and altitude at any point during the run.

The front module communicates with the other computers in the system via ethernet. A CAT-6 RJ45 ethernet cable runs through the mockup sensors, delivery crawler, and the crawler's compound tether. This tether is composed of a flexible Polyethylene Terephthalate sleeving that encloses an ethernet cable, six pneumatic lines used to actuate the delivery crawler pistons, and a taught steel wire to support the crawler's weight when descending and the friction when retrieving. As the crawler moves, the tether will unwind to release the necessary slack. At the end of the run, it will be wound up and retrieved using the system's mechanized reel, as depicted in Figure 7.

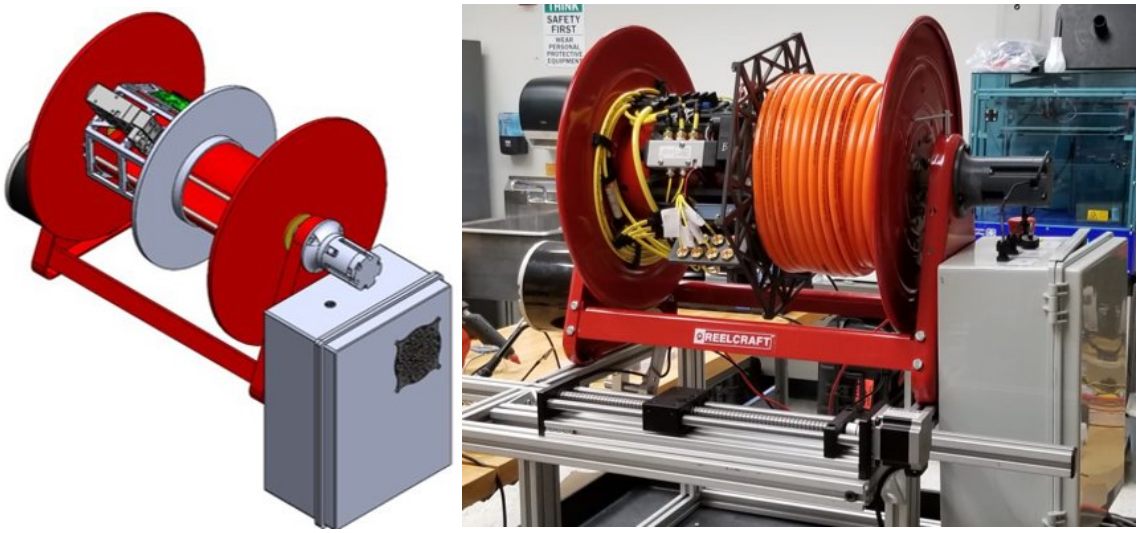

Figure 7. Mechanized Reel, CAD model to the left and prototype to the right

The adapted mechanized reel used for the lateral gamma scanner task is an off-the-shelf mechanical reel. To accommodate the solenoid valves, pistons, relays, crawler control computer, voltage regulators, and ethernet switches that are connected to the tether, half of the spinning body of the reel has been segmented off, as shown in Figure 7. The remaining half of the body winds up the tether. The motor driver, power supplies, and reel's control computer are located in a ventilated control box adjacent to the rotating body, as also shown in Figure 7. This control box features a safety off-button for emergency power cut-off.

To ensure the tether is wound up neatly and with minimal cross-over for safety, the reel is equipped with a wind-assister mechanism. During the first weeks of the summer, the intern developed the system's automation and synchronized it with the reel's motion.

2.2 System Integration

As mentioned in the report's introduction, the internship primarily focused on integrating the system components into an autonomous tool. Due to the system's geometry and the need for modularity, the sensing and actuating functions are divided among separate computers, including micro-computers that run one or more necessary programs. The integration of these programs and computers is crucial to the system's operation, and this is where the Robotic Operating System (ROS2) comes into play.

ROS2 is a framework designed to create and deploy robotic systems. It connects and communicates a group of programs, which can run on a single computer or on a network of computers. This architecture allows developers to distribute a robot's individual components and functions, such as perception, calculation, and actuation, among different devices. This distribution provides modularity and maintainability to the LGS system.

In the LGS system's ROS2 network, individual programs act as nodes connected through communication channels known as topics. Each node can publish information to a topic or subscribe to a topic to receive incoming messages. Topics are used for simple, unilateral communication, where nodes do not require feedback after publishing to a topic. Additionally, ROS2 provides bilateral communication channels called services, which allow a node to submit requests that other nodes can accept and reply to with temporary feedback or a final response. Figure 8 illustrates a ROS2 network with these communication types, which are used by the LGS system. Note that a single node can publish or subscribe to multiple topics or services, as depicted in Figure 8.

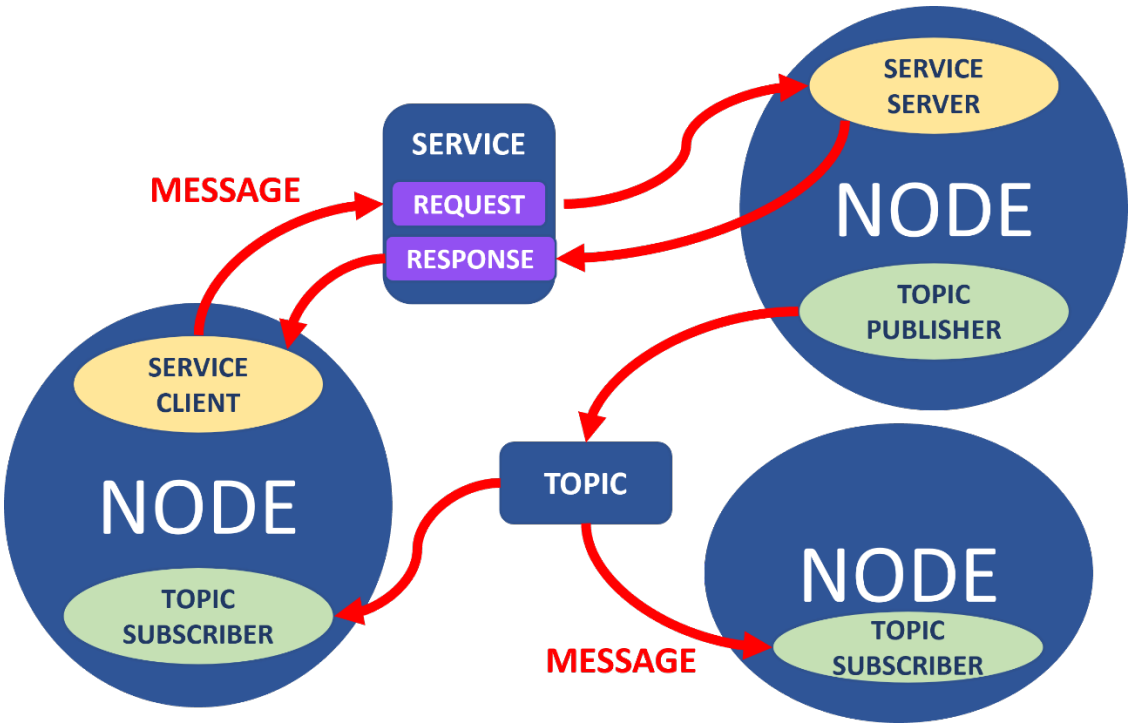

Figure 8. ROS2 nodes communicating unilaterally through a topic and bilaterally through a service

In the LGS communication architecture, two of the main ROS2 programs control the actuation of the pipe crawler's pneumatics and the mechanized reel, which are the two largest mechanical components. Since the winding and unwinding of the tether need to be synchronized with the crawling motion, these nodes must be controlled by a common computer, which oversees the movement synchronization and processing of sensor information through a state machine, described in detail in this report.

To control the pipe crawler and reel, ROS2 services were selected. The central control computer uses services to submit movement requests to the crawler and reel separately. The two single-board computers that control the crawler and reel individually listen for requests made through their respective service. These services allow the reel and pipe crawler to communicate back to the control computer through feedback, enabling the control computer to verify that a requested operation was successful.

Developers can define custom service request messages, which in the LGS system are instructions to the reel and pipe crawler to perform winding/unwinding or different crawl patterns as needed. For the crawler, the service requests consist of a crawl pattern and a signal indicating whether to perform the pattern repeatedly or only once. The crawl pattern specifies the order of activation for the pneumatic pistons that control the crawler's and extender modules gripper, represented in the ROS2 network as an array of six actions.

The custom request allows the control computer to request a variety of sequences for the crawler to perform without having to define them beforehand. For instance, the control computer can request actions such as crawling in reverse, engaging all grippers for a firm lock inside the pipe, or releasing every gripper for retrieval. If needed, these flexible pattern requests can be used to expand the crawler's functionalities in the future.

The ROS2 service for crawler movements also includes a signal that determines whether a crawl is performed a single time or continuously until stopped. This signal is represented in the message as a Boolean value that specifies whether it is true or false. The pipe crawler service replies to the control computer with a count of how many times the action was successfully performed. This count, along with sensor data and the length of the tether that has been unwound, is used by the control computer to estimate how far along the lateral the crawler has advanced, which is crucial for the system's automation.

Similarly, the reel is controlled by a custom action request message that defines an angular velocity value, a time interval between discrete winding pulses, and a Boolean that determines whether the action is performed once or continuously. The reel server node publishes both feedback to the control computer and unilaterally a velocity command to the motor driver node that interfaces with the reel hardware. Figure 9 provides a visual representation of the nodes and their unilateral or bilateral communication.

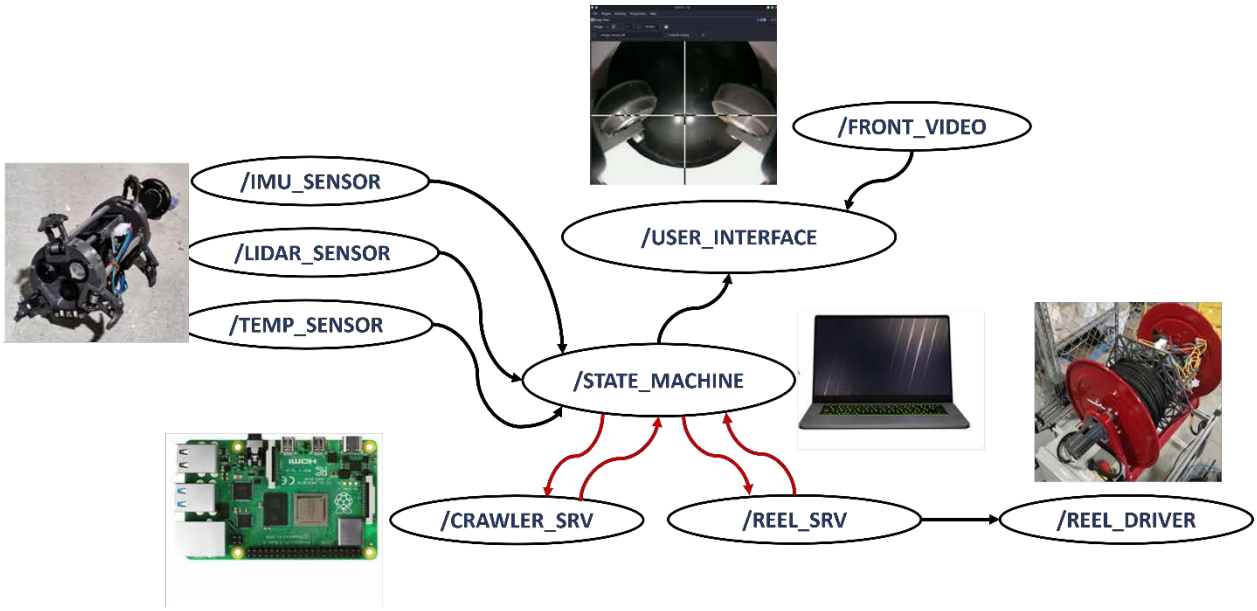

Figure 9. ROS2 node graph; bidirectional services are shown in red, and unidirectional topics in black

The Action interface type provides an important feature that enables cancellation of an in-progress action when a new command is received. This function allows the control computer to send new commands without needing to be aware of the current state of the reel and pipe crawler. For instance, if an error is detected during operation and the crawler needs to be retrieved mid-crawl, the control computer can request the crawler to disengage all grippers and the reel to continuously reel-in the system until the sensors detect a certain altitude.

The sensor information publishing nodes, which run on the embedded Raspberry Pi microcomputer in the front module, constitute the final group of nodes in the network. While the sensory information from these nodes is relayed to the control computer, it can also be done unilaterally or bilaterally. For instance, the camera and environmental sensor continuously and unilaterally publish information through a ROS2 Topic interface. However, other sensors like the IMU and LiDAR are only required at specific times. The control program logic uses these sensors to determine verticality or when the end of the pipe is reached. They are accessed through ROS2 Service interfaces that reply to a request with a response, which could be the IMU's pitch value or the LiDAR's distance value.

As previously mentioned, the ROS2 nodes in the system are connected through a common control node in an external computer. The control computer uses Topics and Services to interface with sensors and actuators that determine the state of the crawler relative to its environment and to act on this information. The State Machine running in the control program makes decisions on what actions to take based on the sensory information.

2.3 Control Architecture

In robotics and automation, state machines are commonly used as a model of behavior. This construct consists of a finite number of defined states, each of which has associated characteristic actions, conditions, and possible transitions to other states. To implement this in our project, we are using PyTransitions, an object-oriented state machine library that interfaces well with ROS2.

By using multiple inheritance, we can instantiate an object that is both a ROS2 node and a PyTransitions state machine.

In our specific application, each state in the machine represents a possible status of the crawler, such as its position in the pipe or whether it has successfully performed an action. The state machine automatically performs actions associated with the current state. For example, when the crawler is in the state representing a long crawl underneath the tank, the state machine requests a crawl forward through the ROS2 network, requests the reel to unwind the respective length of tether to accommodate the crawl, and checks the LiDAR for proximity to the end of the pipe. If the LiDAR service responds with a distance indicating that the end of the pipe is within a determinate distance, the state machine transitions to the state representing a successful crawl, which then sends the respective requests to retrieve the crawler.

Figure 10illustrates the state machine using blocks to represent the states and arrows to represent the possible transitions. To enhance visual clarity, the diagram has been split into an initial and a final half, which is shown in Figure 11. Some transitions between states happen automatically after a scripted action is performed, while others check whether a condition is true or false before allowing transitions. The latter are shown in the diagram with a [condition] symbol after the transition name.

Figure 10. State machine control diagram (beginning half)

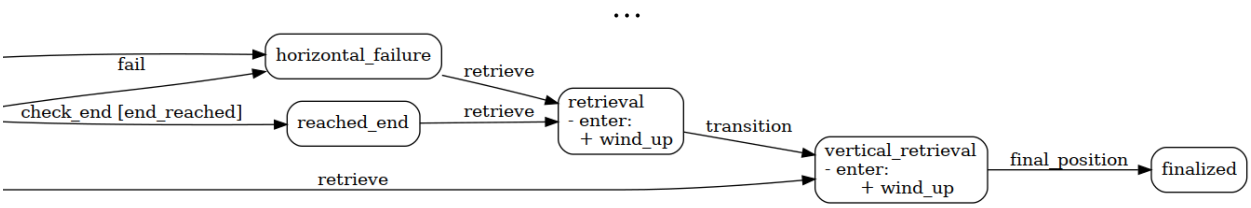

Figure 11. State Machine control diagram (ending half)

The PyTransitions library also offers the flexibility to call functions before or after transitions or within states. These functions can prepare the system for the next state or perform a conditional check before exiting a state to determine the next transition. This is particularly useful in our system to ensure that transitive sections of the operation, such as the elbow that transitions from the vertical drop to the horizontal pipe beneath the tank, are completed successfully before activating the next state.

In our system, a given state can transition into many states depending on the conditions checked by the program's logic. The LGS leverages this feature by assigning most states the ability to transition into a failure state, which has not been implemented yet but is part of the project's future work. When functioning, this transition will be triggered upon an error in the crawler's operation. The failure state will notify the operator about the error and automatically trigger a retrieval state.

To achieve this, we will integrate the linear acceleration readings from the IMU and the odometry calculated using computer vision algorithms. This will help us verify if the commands to crawl forward produce the expected movement of the front crawler. This work will be conducted at the Applied Research Center at Florida International University.

3. TESTING AND RESULTS

During the course of this internship, we conducted separate tests of the individual components in mockup environments that simulated various sections of the crawler's run. In order to test the synchronization of the crawler and the reel using the ROS2 network and PyTransitions library, we ran a short test on a horizontal pipe, as crawling motion is primarily performed on horizontal pipe sections. Requests to check the inclination angle and distance to the end of the pipe, which are used by the control program to determine state transitions, were tested with the front module as an individual unit.

Conducting these modular tests allowed for incremental troubleshooting and development of the system components without assembling the entire pipe crawler. Once individual components were tested, we proceeded to test the entire system at the Hanford Site's Cold Test Facility (Figure 12), which features a life-size tank mockup and a structure tall enough to simulate the vertical section of the pipe. Specifically, a 120-foot vertical pipe was installed with an elbow at the top to feed in the crawler horizontally and another elbow at the bottom connecting the vertical section to the 160-foot horizontal pipe.

Figure 12 . Cold Test Facility Mockup structure

During the internship, the individual components of the pipe crawler were tested separately in mockup environments that simulated different sections of the crawler's run. Specifically, the synchronization of the crawler and the reel using the ROS2 network and the PyTransitions library was tested in a short run of horizontal pipe, while the requests to check the inclination angle and the distance to the end of the pipe were tested with the front module as an individual unit. These modular tests allowed for troubleshooting and development of the system components without the need to assemble the pipe crawler.

Once the components were individually tested, the entire system was tested at the Hanford Site's Cold Test Facility, which has a life-size tank mockup and a structure tall enough to simulate the vertical section of the pipe. The testing included a 120 ft tall vertical pipe with an elbow at the top to feed the crawler horizontally and another elbow at the bottom connecting the vertical section to the 160 ft horizontal pipe.

The tests indicated that the crawler could successfully descend the vertical section by unwinding the reel, transition into the horizontal section and perform the crawl all the way to the end of the pipe, and finally be retrieved by the reel at the top of the mockup. The reel and crawler were synchronized successfully using the ROS2 network and the state machine, and the reel was able to retrieve the crawler with no issues. However, the tests also revealed some issues that will need to be addressed in the future.

The first issue is related to the ethernet connection used to communicate the front module with the rest of the computers in the network. This connection goes through a slip ring that could act as a point of failure. To address this, the intern implemented a break in the system and a transition to a failure state and retrieval in case of a slip ring failure. However, a wireless bridge will be used in the future to bypass the mechanical slip ring that is currently used to maintain connection through a rotating reel.

The second issue encountered was a mechanical failure where the front module would get stuck in between the pipe connections. This issue was caused by the back gripper losing potential grip by folding inwards and losing surface contact area to the inner pipe wall. The crawler would get stuck and when the extenders activated, the back gripper would slide back. This issue will also be addressed in the future.

Finally, to close out the internship, the intern performed a live demonstration of the crawler's capabilities to managers and operating staff at the Cold Test Facility mockup. The demonstration involved the crawler automatically carrying out a representative section of the run, descending through the last section of the vertical pipe by unwinding the reel, triggering the elbow state when the proper inclination was achieved, and crawling through the lateral section before disengaging its grippers and being reeled back up by the reel. The demonstration was purposefully performed in a clear section of pipe for better visibility.

4. CONCLUSION

The objective of advancing the development of the LGS inspection tool was successfully achieved during this internship. The integration of the crawler and reel into the ROS2 network was a major milestone, and a custom message architecture was established to promote modularity. Testing of the front module in a mockup environment was also performed for the first time, with its sensors incorporated into the network as necessary topics and services. The mechanical reliability of the reel and tether was confirmed during testing, as the crawler was successfully retrieved from extended lengths.

Although testing revealed some issues, such as the potential for ethernet disconnection and the grippers folding inwards, these concerns are already being addressed at the Applied Research Center's robotics laboratory. The team will also continue to enhance the capabilities of computer vision and inertial measurement to provide more precise estimates of the crawler's position and to alert the system when mechanical failure hinders the crawler's progress in the pipe.

5. REFERENCES

[1] Open Robotics. Unerstanding-ROS2 Nodes
https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html

[2] Addison Sears-Collins. How to create an action in ROS 2
https://automaticaddison.com/how-to-create-an-action-in-ros-2-galactic-advanced/

[3] Tal Yarkoni, Alexander Neumann. Pytransitions.
https://github.com/pytransitions/transitions

[4] Open Robotics. Synchronous vs Asynchronous Communication
https://docs.ros.org/en/foxy/How-To-Guides/Sync-Vs-Async.html