2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Facile Integration of Robots into Experimental Orchestration at Scientific User Facilities

Chandima Fernando[1], Daniel Olds[1], Stuart I. Campbell[1], Phillip M. Maffettone[1]

***Abstract*— Integration of robots into scientific user facilities, such as the National Synchrotron Light Source II, improves their efficiency and capacity. Many such facilities use the open-source *Bluesky* project for experimental control and orchestration. However, there remains an open challenge in deploying robotic solutions at these facilities that are reconfigurable, extensible, and compatible with pre-existing software infrastructure. Herein, we introduce a framework that uses the Robotic Operating System 2 (ROS2) and *Bluesky* to provide extensible robotic applications, while working under the operational constraints of a large-scale user facility. We demonstrated this framework by integrating a robotic arm to pick and place a sample holder at a beamline, recording a 90% repeatability rate. This provides the groundwork for further new robotics applications at large-scale scientific user facilities that depend on *Bluesky*.**

I. Introduction

Scientific user facilities, such as the National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory (BNL) are state-of-the-art centers that provide advanced equipment and instrumentation to enable cutting-edge research. At the NSLS-II, intense beams of light are generated and used by visiting researchers, called users, for studies of materials spanning a wide range of domains such as physics, chemistry, biology, and material science. Users are granted access to specific instrumentation at the facility *via* a competitive proposal process. Unfortunately, these resources are massively oversubscribed, limiting access and availability for researchers. Moreover, experiments occur in controlled environments under ionizing radiation and cannot be adjusted dynamically without pausing radiation and awaiting safety interlocks. As such, many user facilities have begun exploring robotics and automation to increase efficiency, throughput, and experimental flexibility, hoping to expand access to more users.

The resource users access at light sources during beamtime is called a "beamline", with each beamline offering specialized instrumentation to perform a type of measurement using the X-rays. To ensure safe operations, all beamline experiments are carried out in radiation-proof hutches, and no humans can be inside the hutch when the measurement is ongoing. This limits the capacity for direct human engagement during experiments. Automation is commonplace at beamlines for controlling the optics, sample environments, and detectors [1]. Robotic manipulators have been deployed for limited sample manipulation at a number of beamlines that perform routine experiments with a common form factor [2], [3], [4].

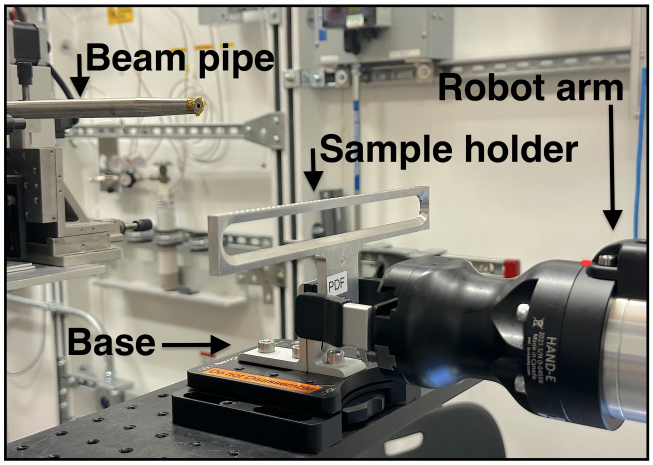

Fig. 1. Robot arm places the sample holder in position at the PDF beamline, a task conventionally performed by direct human manipulation.

Experimental orchestration at a beamline involves concurrent control over many devices, data acquisition, management, and analysis that can feed back into the beamline control. The *Bluesky* project is an open-source Pythonic framework that addresses this orchestration challenge [5]. Every beamline at NSLS-II currently uses aspects of *Bluesky*, with increasing adoption at other light sources and laboratories around the world [6]. Considering this broad adoption by the scientific community, it is crucial to enable facile integration of contemporary robotics solutions into facilities and machinery powered by *Bluesky*.

Nonetheless, there remains an open challenge in deploying robotics solutions that are reconfigurable and extensible at large-scale user facilities. For example, many beamlines currently perform a diverse range of experiments with different form factors. In these cases, a fixed robotic solution akin to a workcell does not sufficiently meet users' needs. We addressed these outstanding needs by developing a generic architecture and deployment pipeline for integrating a Robotic Operating System 2 (ROS2) compatible robot with *Bluesky* experimental orchestration. The main advantage of our approach is that it provides extensibility to new robotic applications while minimizing interruptions to precious operations time. We achieved this by leveraging the contributions of the open-source community (ROS2 and *Bluesky*) to drive development from simulation to laboratory testing, to beamline deployment. This included constructing

*This work was supported by the National Synchrotron Light Source-II, U.S. Department of Energy Office of Science User Facilities, at Brookhaven National Laboratory under Contract No. DE-SC0012704

[1]National Synchrotron Light Source II, Brookhaven National Laboratory, Upton, NY, USA. {wfernando1, dolds, scampbell, pmaffetto}@bnl.gov

a digital and physical simulation environment for prototyping the application. Our architecture made use of reconfigurable ROS2 nodes that integrate with the *Bluesky* orchestration directly. This provided a facile robot deployment at the beamline, for plug-and-play integration with their existing software stack.

Our key contributions are as follows:

- A generic architecture for integrating ROS2 robots into the *Bluesky* project for experimental orchestration.
- Demonstrating a sample manipulation task with *Bluesky* and ROS2 using a digital and tabletop simulation.
- Concurrent orchestration of a UR3e in an active beamline, using the existing facility software stack.

II. Related Work

Physical and life science researchers are increasingly automating experiments for accelerating research with higher throughput, repeatability, precision, and safety [7]. Robotic arms and mobile manipulators are often employed to add flexibility and integration to existing scientific workflows [8], [9], [10], [11], [12], [13]. A recent perspective asserts that "[self-driving labs] provide an exciting semi-structured environment where the robotics community can transfer their methods to novel applications" [7]. Despite these many advancements, there are outstanding needs in designing modular and flexible integration of anthropomorphic robots, with democratized access to the scientific community.

Of particular interest for laboratory automation is object manipulation, e.g., pick-and-place. This is a well-developed industrial application for picking and placing items on a conveyor belt. A series of FlexPicker robots that pick and place salami on mini-pepperoni, Selective Compliance Assembly Robot Arm (SCARA) robots that pick and place machine parts, and Delta robots that pack food items from a conveyor belt are such instances [14], [15]. An extension to this technology has been to selectively pick and place items from warehouse shelves to autonomously fulfill customer orders [16]. Minimalistic end effectors further improve efficiency by picking and optimally placing differently shaped items in a box [17]. Precision agriculture is another application area where the pick-and-place action is used [18], [19]. Here, we build on past works to construct a useful demonstration of our integration.

The question of integration in a laboratory setting has seen a rise in recent attention. ARChemist is a workflow management program for a laboratory conducting chemistry-related research [9]. The main features of ARChemist are the separation of workflow components and the introduction of the YAML files for easy re-configuration. The separately managed workflow components are responsible for the state of the whole experiment floor, managing the data storage, and the integration of different robotic hardware modules through ROS2. ChemUP is a similar architecture that was developed with the intention of automating the workflow in a chemistry laboratory [20], albeit does not support integrating multiple robotic systems into the workflow. Vescovi *et. al* provided a substantial advancement in modular architectures called WEI, by building on the ideas of workcells, actions, and workflows [21]. This provided a generic set of abstractions meant to be used in coordinating unit operations in all domains of physical and life sciences. While these tools provide substantial flexibility, the former are limited in their scope and leave little opportunity to integrate with existing experimental orchestration stacks. On the other hand, the innovations provided by WEI are flexible in higher-level coordination, but pay no attention to the internal orchestration of a given unit operation (in this case, a beamline).

The *Bluesky* project is a community-driven Python project for experimental orchestration used in the software stack of beamlines around the world [5]. It contains libraries for hardware integration (`Ophyd`), specification and orchestration (`Bluesky`), online visualization and analysis, data access, and real-time autonomous experimentation. Redundantly named, `Bluesky` is also a library within the *Bluesky Project* dedicated to experiment control and data collection. This library maintains the `RunEngine` which orchestrates the execution of experimental plans. To date, robotic integration into the *Bluesky* stack has been completed on a case-by-case basis, often through the hardware control layer present at a beamline [2], [4]. Control layers such as the Experimental Physics and Industrial Control System (EPICS) are common to beamlines, and thus have existing `Ophyd` integration through the use of process variables. However, using EPICS for robot control forces any complicated robotic processes to be reduced to primitive types (integers or strings) to be set and read, limiting the orchestration process from accessing any dynamic metadata.

Bluesky applications have been embedded with artificial intelligence to produce self-driving experiments [1], [22], [23], albeit sample management and manipulation (i.e., loading and unloading) is often a limiting requirement. At beamlines with a common form factor across all experiments, anthropomorphic robots have been deployed to address this, integrating the beamline as a single workcell [2], [3], [4]. However, at user facilities where the composition of a unit operation changes between experiments—or even throughout a single experiment—this approach fails to meet the requirements of flexibility required by user-driven science. An example of this that we use to commission our application occurs at the Pair Distribution Function (PDF) beamline at NSLS-II, where a user may wish to measure several isolated materials and then a series of devices composed of composites of those materials.

A unique challenge associated with user facilities lies in the secure and robust deployment of new technologies. To this end, NSLS-II maintains a routable network with VLAN isolation, zero-trust security, and upward of 1000 machines provisioned using Red Hat Enterprise Linux (RHEL) [24]. These machines share a common network file system with users' home directories and redundant central storage for scientific data. While this approach enables the particle accelerator and beamlines to operate securely, in real-time, using virtualization, it creates certain barriers to deploying robotic solutions using ROS2, where not all applications will

be supported on RHEL. Containerization has been employed at user facilities elsewhere for research software [25], and we sought to use a combination of virtualization for network isolation and resource allocation with containerization for environment management.

III. Design

A. Overview

Our generic approach was developed in the context of a commissioning experiment. At the PDF beamline at NSLS-II, different form factors of samples and holders are loaded onto brackets with magnetic kinematic mounts (Thorlabs). Our core objective was to design a system that can be deployed at PDF to load samples from temporary storage into the experimental environment. In this case, a UR3e 6-dof arm was mounted on a 2-axis stage to remove it from direct exposure to incident X-rays during the measurement. This design required a manipulator that could controlled by the *Bluesky* `RunEngine` concurrently with the other devices at the beamline. Lastly, it was crucial that the approach could be rapidly reparameterized on installation into the beamline, to minimize down-time when the X-ray beam was available.

We approached this design challenge using the *Bluesky* and ROS software stacks. We built a parameterized solution that was developed in a digital simulator, and then tested in a simulated environment (with the robot in a laboratory) prior to beamline deployment. Below we describe these designs in conjunction with the infrastructure and deployment strategies.

B. Bluesky and Ophyd

We developed an interface for the rapid prototyping and deployment of robotic solutions that were compatible with *Bluesky*. Specifically, this work focused on seamless integration with *Ophyd*. *Ophyd* is used to abstract the details of a vast diversity of hardware (e.g., motors, temperature controllers, optics, detectors, and sensors) into a unified high-level interface with methods like `trigger`, `read`, and `set`. Commonly implemented on top of the EPICS for communication, *Ophyd* has also been used in conjunction with serial and TCP/IP communication. Herein, we designed *Ophyd* devices to use the ROS client library for communicating with robots.

Most methods in *Ophyd* will return a `Status` object that is a derivative of a `Future` for multi-threaded and asynchronous applications. This enables the `RunEngine` (akin to the ROS2 executor) to orchestrate complex plans of motion, actuation, and detection for an experiment. Because a ROS2 `Action` will return a `Future`, we update the corresponding *Ophyd* `Status` using callbacks during the feedback and completion stages of executing a goal in a ROS2 `Action`. Specifically, we construct an `ActionStatus` that inherits from *Ophyd* `DeviceStatus`, and an `ActionMoveable` that inherits from ROS2 `Node` and *Bluesky* `Movable`. The `ActionMoveable` uses the *Ophyd/Bluesky* `set` interface to trigger an `Action` by setting a goal and returns an `ActionStatus` that is subscribed to the `Future` returned by the ROS2 `Node`. We then use the `RunEngine` to orchestrate the robot action in concert with the existing equipment at the beamline.[1]

By exposing ROS2 directly to *Bluesky* in this way, we reduce complexity and create opportunities for rich user interfaces and control. This enables us to exploit the ROS2 ecosystem for robotic applications, as well as allowing for dynamic communication between the thread executors in a robot service (`Action`) and in the beamline service (`RunEngine`). Furthermore, the *Bluesky* project contains many existing and ongoing developments for experiment planning and monitoring, such as progress bars, data visualization, and queue-based workflow management [23]. Since these interfaces frequently depend on `Status` objects, we focused our design on ensuring the exposure of co-routine status information (*i.e.,* by a `Action` interface).

C. ROS2 tools

In this work, we used ROS2 Humble and the associated Universal Robots ROS2 drivers.[2]

1) Action sever: ROS2 `Action` servers are ideal candidates for managing the sample movement task with *Bluesky*. They accept or reject a requested task, provide continuous feedback on task execution, and can cancel a task in execution. Figure 2 shows the command flow of the action server framework. The `RunEngine` executes a plan that includes a 'goal request' for the server, using the same syntax *Bluesky* uses to adjust devices. The goal request contains the object name to be picked, the starting location, the stopping location, and the type of task. We categorized tasks into 4 types: pick up, drop off, return pick up, and return drop off. Each represents a sequence of sub-tasks to perform. This approach can be trivially extended to include multiple storage locations. Upon receiving a goal, the task planner selects the appropriate sequence of sub-tasks. The sub-tasks consist of a combination of a Cartesian movement, joint movement, and/or gripper control. Cartesian and joint movements of the robot are achieved through MoveIt! and MoveIt! Task Constructor (MTC) respectively.

2) MoveIt! and the Task Constructor: We chose the ideal path planning strategy between MoveIt! and MTC depending on the motion required. We aimed to generate simple movements that fit the physical constraints of the environment and simplify deployment at the beamline. For instance, sample holders are constrained to being upright throughout each task as any significant change in their orientation would result in materials being lost or inaccurately positioned. This required us to maintain the gripper orientation after grasping the sample holder. Therefore we used Cartesian movement from the move-group interface provided by MoveIt! to approach and retreat the sample holder from the magnetic bases. Once the sample holder is firmly grasped by the gripper, it needs to be moved to in front of the beamline to conduct experiments.

[1]Ongoing development at https://github.com/maffettone/erobs

[2]https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver

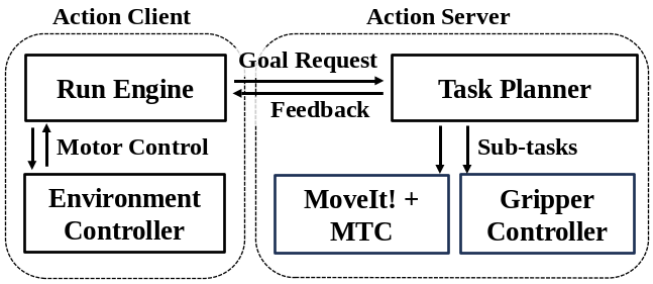

Fig. 2. Graphic representation of command flow in the ROS2-Bluesky interface. The `Action` client is controlled by a `RunEngine` that operates the rest of the beamline machinery. The client issues goal requests to the ROS2 `Action` server that then controls the robot application while providing feedback to the `RunEngine` through the client.

The robotic arm performs the required rotation over the base joint using MTC. We use MTC to execute specific joint movements one at a time, and the move group to plan its trajectory provided a target pose.

D. Re-configuration

To meet the requirements of rapid deployment for diverse environments and tasks, our framework was designed to be reconfigurable. We use a YAML file to parameterize the location of the sample holder, its destination, values specific to the robot arm, and the obstacles in the environment. These parameters are human-readable and can be adjusted by the ROS2 parameter server at runtime. Secondly, we constructed a development pipeline that included a digital simulator, and a physical simulated environment, before our beamline deployment. This enabled our team to contribute to the control package without consistent access to the robot. Furthermore, it let us "pressure test" the rapid reconfiguration in a low-stakes environment that approximated the beamline on a tabletop.

E. Infrastructure and Deployment

Our deployment strategy was subject to the resource constraints associated with large scientific user facilities. Principally, the opportunity to test a robotic application at a beamline before a user experiment is limited to the periods when the facility is shut down (up to 30 days quarterly) or undergoing accelerator studies (a few weekends per month). As an advanced scientific user facility, there are also substantial cyber security concerns to maintain confidentiality, integrity, availability, reputation, and compliance.[3] To overcome these challenges, we built our tools as portable services with common networking configurations in development, laboratory testing, and beamline deployment.

The network at NSLS-II has a 4x100 Gbps connection to the facility, 2x100 Gbps connection at the testing lab, and a 4x100 Gbps connection from BNL to the outside world. To provide a consistent and secure computing environment, all machines are centrally provisioned with Red Hat Enterprise Linux (RHEL) using a Red Hat Satellite Server with configuration managed by Ansible.

[3]https://www.directives.doe.gov/terms_definitions/cybersecurity

NSLS-II makes use of extensive virtualization using VMWare vSphere clusters, including the real-time operating systems that control the particle accelerator. We provisioned VMs with network interfaces to isolated VLANs at the testing lab and beamline, and assigned IP addresses in both segments according to a common convention for subnets. This enabled a facile transition between the physical simulator and the beamline that maintained security and minimized operational interruptions.

Lastly, we deployed our robot drivers and services using containers that shared the host VM network interfaces. We pursued containerization to overcome the challenges of second-tier support for RHEL by ROS2, complexities introduced by beamline-specific environments for operations, and difficulties in maintaining common development environments across distributed teams. Each service necessary to run the UR3e (robot driver, gripper driver, MoveIt! move group, and ROS2 action server) was assembled into an independent Ubuntu container on a RHEL 8.8 VM that was isolated from scientific data and sensitive instrumentation. A separate VM used for normal beamline operations was used to deploy a ROS2 container which included *Bluesky* and EPICS. This container mounted the Python environments necessary for operating the beamline and any necessary private configurations so that the `ActionMoveable` clients could be used concurrently with existing beamline orchestration.

IV. Simulation Evaluation

A. Terminology

In typical measurements at the PDF beamline, the X-ray beam is directed at a sample confined inside a sealed capillary tube. Each individual measurement of a sample takes on average between 1 second and 5 minutes (depending on the sample under study), though samples are often measured multiple times to improve data quality or track material changes over time. A series of samples are loaded into a sample holder, and the sample holder is placed upright and attached to a mobile base with a kinematic mount. Throughout a measurement, the mobile sample stage or base is moved to adjust which capillary tube is currently in the 400 *um* x 400 *um* X-ray beam. The place where we store the samples before and after an experiment is called storage, and the location where the sample is placed in front of the beam pipe is called in-beam. We segmented the pick-and-place task of a sample holder into four sub-tasks: pickup, place, return pickup, and return place. In the pickup task, we remove the sample holder from the storage, and in the place task, we move it to the base at the in-beam location. After the experiment is complete, return pickup and return place tasks would bring the sample holder from the in-beam location back to the storage.

B. Simulation Setup

To construct a digital simulation we used the URSim simulator from the manufacturers of the UR3e to mimic the physical robot and the state of the physical robotic

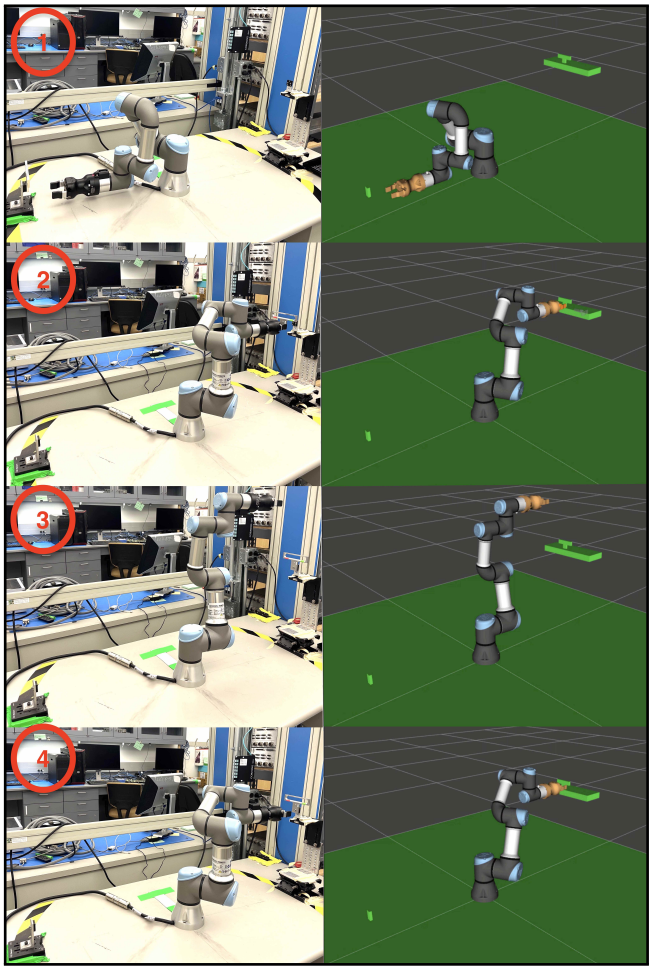

Fig. 3. The sequence of robot motion at the lab experiments and their respective simulation on RViz are shown in the four images from top to bottom. 1) The robot is at the pre-pickup stage before picking up the sample holder 2) The robot places the sample holder at the inbeam 3) The robot is at the rest position 4) The sample is picked up to return.

system. We used the RViz tool for ROS2 to visualize the planned motions, environment obstacles, and the robot status. Furthermore, we extended the UR robot description file to include a Robotiq Hand-E gripper.

We paired this digital simulation first with a tabletop approximation of the beamline environment. This physical simulator provided a test bed for the pick-and-place task and the deployment infrastructure. The obstacles in the RViz visualization and mock environment approximated the real obstacles in the beamline environment [Fig. 3].

We used the YAML parameter files to account for the obstacle space inside a beamline environment, which can vary during each experiment and between each beamline. At the beamline, the robot base will be translated in and out of incident beam to protect the experiment and robot. To coordinate this, we projected all the obstacles and objects into a single reference volume for the required tasks. This traded off an increased complexity of path planning, for a decreased complexity in broadcasting beamline motors to the ROS2 transform service.

TABLE I
SUCCESS RATE TEST RESULTS

Experiment Setup	Number of Attempts	Number of Successes	Failures Due to Positioning	Success Rate
Tabletop Sim	30	23	6	77%
Beamline	30	27	1	90%

C. Simulation Results

We used the digital simulation to test the interface between *Bluesky* and ROS, such that the `RunEngine` could execute the ROS2 `Action` inside a *Bluesky* plan. A *Bluesky* plan is a coroutine that is implemented as a Python generator. A typical plan is sketched out below. The total simulation time is 2 minutes and 56 seconds, and the sequence of robot movement is shown in Figure 3. We did not simulate the translation of the robot base, because the precision of the 2-axis stage is high enough to expect the robot to return to the same position.

```
def plan(node, stage, detector):
    yield from bps.abs_set(node, pickup_args,
        wait=True)
    yield from bps.abs_set(node, place_args,
        wait=True)
    yield from bps.mv(motor, safe_position)
    yield from bps.trigger_and_read(detector)
    yield from bps.mv(motor, load_position)
    yield from bps.abs_set(node,
        return_pickup_args, wait=True)
    yield from bps.abs_set(node,
        return_place_args, wait=True)

node = PickPlaceDevice(...)
RE = RunEngine(...)
RE(plan(node, motor, detector))
```

V. EXPERIMENTAL EVALUATION

For the physical simulation and deployment at the PDF beamline, we define a successful operation to include: i) the robot starting from the rest position; ii) grasping the sample holder; iii) moving the sample holder to the in-beam location; iv) returning the sample holder from the in-beam location to the storage; v) and moving to the rest position. In the beamline deployment, a successful operation includes an additional task of moving the robot base out of the incident beam and back. Furthermore, we focus our measurement of success on the *Bluesky* interface's ability to control the robot in the mock environment and to coordinate the motion of the 2-axis stage. Therefore, all control was accomplished using a single `RunEngine` and the existing beamline infrastructure and configurations. We considered three figures of merit to evaluate our robot solution, the success rate of the complete task described above, the experimental throughput that was measured by the exchanged samples per hour, and potential person-hour savings.

A. Testing in a physical simulation

In our tests at the mock tabletop environment, we successfully demonstrated *Bluesky* control of the robot and pick-and-place completion. Here, we ran all drivers and services in containers on a single VM. Figure 3 shows the

actions proceeding alongside their visualization in RViz. In a single repeatability test over 30 repeat tasks, the robot had a 77% success rate with six failures due to poor positioning (the receiving end of the storage mount became displaced, resulting in a failure to return to storage). The remaining error involved an operator manually interrupting the test. The positioning errors were ameliorated in the beamline deployment by refining the parameters of object location, and rigidly affixing the storage mount. Due to the limited access to the beamline, we addressed this for the deployment rather than the physical simulator.

This simulation differed slightly from the beamline deployment in several ways. As mentioned, the exact object and obstacle location differed between the mock environment and the beamline. Secondly, there were no additional motors to control in the simulation using *Bluesky*. Thirdly, we provisioned all services on a single VM in the simulation, using the host networking. Because of this, there was a difference in the behavior of dynamic discovery of ROS2 nodes between the simulator and the beamline.

B. Deployment at PDF

We installed the robot at the beamline on the 2-axis stage and successfully completed the sample manipulation. We controlled the beamline devices (including the stage for translating the robot base) and the robot using a single *Bluesky* plan. Figure 4 shows the sequence of sample pick-and-place. Following the movement of the sample holder to in-beam from storage, the robot pose was then set to be upright in preparation to be translated out of the incident beam so that the experiment could proceed. This rest position was chosen to prevent the robot from colliding with objects in the environment when translated. We programmed the stage to move 26 cm from the current position. A complete cycle of pick-and-place and movement took 4 minutes and 30 seconds. The stage is returned to the initial position following the beamline experiment and the robot arm successfully returned the sample holder from the in-beam location back to the storage.

Our repeatability tests at the beamline were conducted for 30 attempts and the results are recorded below. We recorded a 90% success rate in completing the full task. The failures included a missed sample pickup, force trigger of a protective stop, and network connectivity loss. These are tractable errors that can be solved with parameter refinement, limiting the acceleration of the `move_group`, and running containers in persistent processes. In comparison to the physical simulation results in Subsection V-A, there was only one failure that was due to positioning inaccuracies. The two other failures resulted from losing the connectivity to the robot and the trajectory manager in MoveIt! aborting the path planning.

Time for a full sample change with the robot, excluding the time of the unoptimized platform shift, took ∼200 seconds, whereas an ideal operator can pause operation and change the sample in ∼90 seconds. Practically, however, 3-shift dedicated operators to change samples 24 hours a day are not available, and a real user exchanges 8–9 samples per hour on average. This represents a potential 2-fold increase in sample throughput through the use of automation prior to optimization, and could at minimum save 12–16 hours/day in labor associated with 2nd and 3rd shifts. This increases the number and kind of experiments that a beamline can consider.

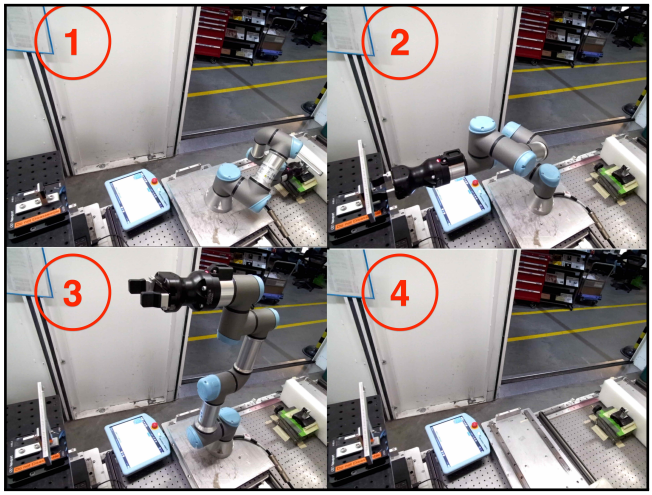

Fig. 4. Four of the key stages of the pick-and-place action at the beamline. 1) Robot picks up the sample holder. 2) Sample holder is placed at in-beam. 3) Robot is ready to be moved. 4) Platform with the robot is shifted using *Bluesky*.

VI. Conclusions and Future Work

In this work, we presented a generic architecture for integrating ROS2 applications into the *Bluesky* stack at scientific user facilities. We developed a pipeline for digital and physical simulations of robotic applications to work with significant access constraints and subsequently deployed a robot for sample manipulation at the PDF beamline. Flexible robotic applications are crucial for making efficient use of limited beamtime and enabling self-driving beamlines [6]. Using this architecture and pipeline, we can explore diverse robotic applications at beamlines without augmenting their existing software infrastructure. This approach to flexible automation has been extended to sample form factors, and can be extended to other *Bluesky* enabled beamlines or even to self-driving laboratory applications. Towards this overarching goal of providing, we intend to extend this work to exploit more *Bluesky* functionality, increase operational repeatability, and incorporate a larger library of tasks.

Acknowledgements

The authors thank Max Rakitin for his insights regarding *Bluesky* and *Ophyd*. This research is supported in part by BNL Laboratory Directed Research and Development (LDRD) Grant No. 23-039, "Extensible robotic beamline scientist for self-driving total scattering studies." This research used resources of the PDF (28-ID-1) Beamline at NSLS-II.

REFERENCES

[1] A. Barbour, S. Campbell, T. Caswell, M. Fukuto, M. Hanwell, A. Kiss, T. Konstantinova, R. Laasch, P. Maffettone, B. Ravel, *et al.*, "Advancing discovery with artificial intelligence and machine learning at nsls-ii," *Synchrotron Radiation News*, vol. 35, no. 4, pp. 44–50, 2022.

[2] E. O. Lazo, S. Antonelli, J. Aishima, H. J. Bernstein, D. Bhogadi, M. R. Fuchs, N. Guichard, S. McSweeney, S. Myers, K. Qian, *et al.*, "Robotic sample changers for macromolecular x-ray crystallography and biological small-angle x-ray scattering at the national synchrotron light source ii," *Journal of synchrotron radiation*, vol. 28, no. 5, pp. 1649–1661, 2021.

[3] A. R. Round, D. Franke, S. Moritz, R. Huchler, M. Fritsche, D. Malthan, R. Kläring, D. I. Svergun, and M. Roessle, "Automated sample-changing robot for solution scattering experiments at the embl hamburg saxs station x33," *Journal of Applied Crystallography*, vol. 41, no. 5, pp. 913–917, 2008.

[4] D. Y. Ozgulbas, D. Jensen Jr, R. Butler, R. Vescovi, I. T. Foster, M. Irvin, Y. Nakaye, M. Chu, E. M. Dufresne, S. Seifert, *et al.*, "Robotic pendant drop: containerless liquid for μs-resolved, ai-executable xpcs," *Light: Science & Applications*, vol. 12, no. 1, p. 196, 2023.

[5] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's ahead: A multi-facility collaboration for an a la carte software project for data acquisition and management," *Synchrotron Radiation News*, vol. 32, no. 3, pp. 19–22, 2019.

[6] P. M. Maffettone, S. Campbell, M. D. Hanwell, S. Wilkins, and D. Olds, "Delivering real-time multi-modal materials analysis with enterprise beamlines," *Cell Reports Physical Science*, vol. 3, no. 11, p. 101112, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666386422004143

[7] P. M. Maffettone, P. Friederich, S. G. Baird, B. Blaiszik, K. A. Brown, S. I. Campbell, O. A. Cohen, T. Collins, R. L. Davis, I. T. Foster, *et al.*, "What is missing in autonomous discovery: Open challenges for the community," *arXiv preprint arXiv:2304.11120*, 2023.

[8] B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, *et al.*, "A mobile robotic chemist," *Nature*, vol. 583, no. 7815, pp. 237–241, 2020.

[9] H. Fakhruldeen, G. Pizzuto, J. Glowacki, and A. I. Cooper, "Archemist: Autonomous robotic chemistry system architecture," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6013–6019.

[10] P. Shiri, V. Lai, T. Zepel, D. Griffin, J. Reifman, S. Clark, S. Grunert, L. P. Yunker, S. Steiner, H. Situ, *et al.*, "Automated solubility screening platform using computer vision," *Iscience*, vol. 24, no. 3, 2021.

[11] G. Pizzuto, J. De Berardinis, L. Longley, H. Fakhruldeen, and A. I. Cooper, "Solis: Autonomous solubility screening using deep neural networks," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[12] J. X.-Y. Lim, D. Leow, Q.-C. Pham, and C.-H. Tan, "Development of a robotic system for automatic organic chemistry synthesis," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 2185–2190, 2020.

[13] B. P. MacLeod, F. G. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. Yunker, M. B. Rooney, J. R. Deeth, *et al.*, "Self-driving laboratory for accelerated discovery of thin-film materials," *Science Advances*, vol. 6, no. 20, p. eaaz8867, 2020.

[14] Z. H. Khan, A. Khalid, and J. Iqbal, "Towards realizing robotic potential in future intelligent food manufacturing systems," *Innovative food science & emerging technologies*, vol. 48, pp. 11–24, 2018.

[15] S. D. Han, S. W. Feng, and J. Yu, "Toward fast and optimal robotic pick-and-place on a moving conveyor," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 446–453, 2019.

[16] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2016.

[17] R. Shome, W. N. Tang, C. Song, C. Mitash, H. Kourtev, J. Yu, A. Boularias, and K. E. Bekris, "Towards robust product packing with a minimalistic end-effector," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9007–9013.

[18] H. Wang, Y. Lin, X. Xu, Z. Chen, Z. Wu, and Y. Tang, "A study on long-close distance coordination control strategy for litchi picking," *Agronomy*, vol. 12, no. 7, p. 1520, 2022.

[19] K. Rahul, H. Raheman, and V. Paradkar, "Design and development of a 5r 2dof parallel robot arm for handling paper pot seedlings in a vegetable transplanter," *Computers and Electronics in Agriculture*, vol. 166, p. 105014, 2019.

[20] A. J. Hammer, A. I. Leonov, N. L. Bell, and L. Cronin, "Chemputation and the standardization of chemical informatics," *JACS Au*, vol. 1, no. 10, pp. 1572–1587, 2021.

[21] R. Vescovi, T. Ginsburg, K. Hippe, D. Ozgulbas, C. Stone, A. Stroka, R. Butler, B. Blaiszik, T. Brettin, K. Chard, *et al.*, "Towards a modular architecture for science factories," *arXiv preprint arXiv:2308.09793*, 2023.

[22] J. Hill, S. Campbell, G. Carini, Y.-C. K. Chen-Wiegart, Y. Chu, A. Fluerasu, M. Fukuto, M. Idir, J. Jakoncic, I. Jarrige, *et al.*, "Future trends in synchrotron science at nsls-ii," *Journal of Physics: Condensed Matter*, vol. 32, no. 37, p. 374008, 2020.

[23] P. M. Maffettone, D. B. Allan, S. I. Campbell, M. R. Carbone, T. A. Caswell, B. L. DeCost, D. Gavrilov, M. D. Hanwell, H. Joress, J. Lynch, B. Ravel, S. B. Wilkins, J. Wlodek, and D. Olds, "Self-driving multimodal studies at user facilities," 2023.

[24] M. Rakitin, S. Campbell, D. Allan, T. Caswell, D. Gavrilov, M. Hanwell, and S. Wilkins, "Next generation experimental data access at nsls-ii," in *Journal of Physics: Conference Series*, vol. 2380, no. 1. IOP Publishing, 2022, p. 012100.

[25] A. Malviya-Thakur, D. E. Bernholdt, W. F. Godoy, G. R. Watson, M. Doucet, M. A. Coletti, D. M. Rogers, M. McDonnell, J. J. Billings, and B. Maccabe, "Research software engineering at oak ridge national laboratory," *Computing in Science & Engineering*, 2023.