ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP013

STATUS ON CONTINUOUS SCANS AT BESSY II

N. Greve*, G. Pfeiffer, M. Neu, D. Kraft, M. Brendike†
Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, Berlin, Germany

Abstract

Continuous energy scanning is an important feature for many beamlines at BESSY II. In 2015 this method was used at 11 undulator and 6 dipole beamlines. Since then the demand for this feature – especially among new build beamlines – increased, while the availability of the used hardware decreased. To tackle this problem, we investigate alternative hardware and software solutions. By introducing an independent high level controller between the device controllers we can compensate for communication incompatibilities and hence increase flexibility.

This paper shows the status of our research. The ideas leading to a first prototype, the prototype itself, and first results will be presented.

INTRODUCTION

The synchrotron light source BESSY II supplies some 35 beamlines with brilliant soft x-rays from undulator and dipole sources with defined photon energy ranges. In this context, scanning across a provided photon energy range by synchronizing the movement of multiple actuators along a beamline without intermediate stops is referred to as continuous scanning (CS) or the continuous mode (CM) [1]. CS significantly decreases the time required for energy scanning experiments like x-ray absorption spectroscopy. Additionally it reduces sample exposure and hence increases sample lifetime, reduces optomechanical vibrations due to smoother acceleration and deceleration phases, and increases a beamline's experimental portfolio by including time-resolved experiments [2–4]. The above points combined with its wide use at BESSY II and other synchrotron facilities make CS an essential technique for today and future beamline operation.

The methods behind current CS at BESSY II beamlines are explained in detail by Balzer *et al.* [1]. The motion controllers (VME-based hardware) of the monochromator and the undulator communicate via a dedicated CAN Bus interface. The monochromator input-output-controller (MONO IOC) holds the gap-to-energy look-up-table (LUT) and can request the undulator to move from gap position A to gap position B. During this move the undulator will report its current gap position regularly via the CAN-bus to the MONO IOC. The monochromator then adjusts it's optical elements according to the reported gap position and the stored LUT. The undulator moves with a symmetrical trapezoidal gap velocity profile with a constant maximum speed. This is equivalent to moving from energy X to energy Y with a non constant rate of change of photon energy. Sampling this energy at regular intervals – which is often wanted – is thus more complex since the triggering needs to be done by additional hardware like PandA Box instead of a fixed time based sample rate [5].

Performing CS as described has limitations. First, scanning with a constant rate of energy change is not supported. This could be changed by implementing a trajectory generator on the MONO IOC, but this is challenging as the computational resources on the VME hardware are very limited. Second, current CS depends on the use of specific near-end-of-life hardware, which can lead to support and incompatibility issues in the future. To mitigate this risk other institutes also start searching for alternatives hardware solutions [6, 7].

Off-the-shelf motion controllers are available to synchronize movement of multiple motor types. Integrating them into the existing BESSY II beamline control environment on the other hand is challenging. Their proprietary closed source nature often reduces their configurability, which makes it difficult and sometimes even impossible to integrate them with existing in-house applications. Hence a freely configurable solution, which can be integrated with existing applications is necessary.

In this paper we present such a solution in the form of a prototype. The following chapters discuss the prototype's design and implementation in more detail. We finish with first results from tests performed on real beamline equipment outside of BESSY II user operation.

PROTOTYPE DESIGN

The main goal of this project is to move the undulator and monochromator axes synchronously using variable speed over time.

For the beginning we focused on controlling the movement of the undulator gap and the monochromator grating and mirror. Controlling the undulator shift is subject of future iterations.

Designing a modular software architecture is another requirement. Beamlines often vary in their configuration in regard to the type of monochromator and undulator present at the beamline, which makes a modular software design necessary. In a modular software design it is possible to include and exclude modules without breaking the whole software architecture. This makes it possible to combine several software modules, based on the beamline configuration at hand.

Additional requirements were:

- to further reduce implementation costs, this first prototype is implemented as an open-loop system, but extending the system to become a closed-loop system has to be possible;

* nico.greve@proton.me
† maxim.brendike@helmholtz-berlin.de

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP013

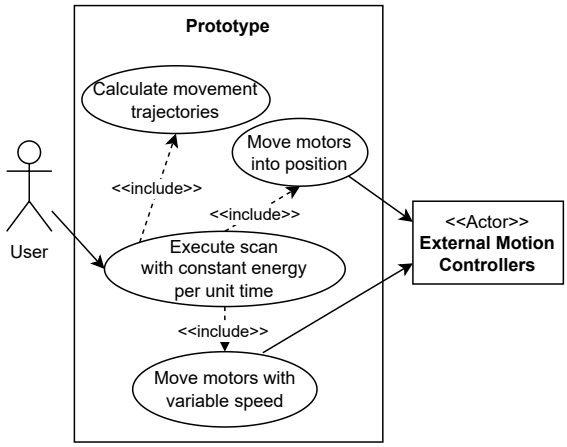

Figure 1: Simplified use case diagram.

- do not introduce new communication protocols, work with what is already there;
- find and patch current limitations in the existing API of the monochromator and undulator motion controller.

Use Cases

Moving the monochromator and undulator synchronously hints at the main use case for the prototype. This is shown in Fig. 1.

The user will command the prototype to execute a scan of a specific energy range with constant rate of energy change. Based on the scan parameters included in the user command, the prototype will calculate the movement trajectories for the undulator and monochromator axes. The starting points of the trajectories are then used to command all connected motion controllers to move their axes into start position. When in start position, the prototype will perform a synchronous movement along the calculated trajectories. All connected motion controllers receive velocity update commands until they reach the end of the trajectory. The developed system is not directly connected to any motors, but communicates with the motion controllers which execute the motor movement.

IMPLEMENTATION

The ideas discussed in the previous section lead to the presented implementations. After a short introduction of the used hardware, the focus switches mainly on the software side.

Hardware

For the prototype a Raspberry Pi 3B+ was used. To communicate with the undulator and monochromator motion controller (Unidrive and Beckhoff respectively), two Digilent Pmod CAN (Revision B) were connected to the Raspberry Pi via SPI. This type of CAN extensions board uses the Microchip MCP25625 controller with an integrated transceiver [8].

Linux comes with a driver for that type controller [9]. For the prototype the server version of Ubuntu 20.04.5 LTS was used. After configuring the device tree overlay, the CAN interfaces can be used like any other network interface.

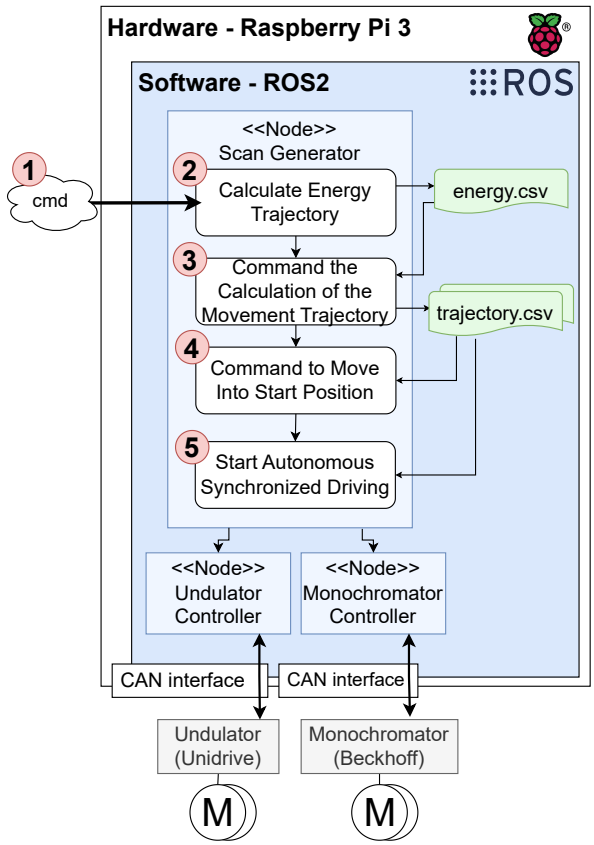

Figure 2: Simplified system architecture [11, 12].

Software

Based on the previously discussed necessities for modularity, the ROS2 software framework was chosen. ROS stands for "Robot Operating System" and is an open source software framework developed to built robot applications. During development ROS2 version "Foxy Fitzroy" was used. ROS2 libraries are available for C++ and Python. Third-party libraries can be included as usual.

In the context of ROS2, software modules are called Nodes. Nodes can communicate using three different communication models. They can communicate using remote procedure calls called *Services*, by using a publish-subscribe messaging model using *Topics* or by a concept named *Actions* [10].

During development the *Services* model was exclusively used. By splitting the software into multiple nodes, ROS2 encourages the software architecture principle of *separation of concerns* which leads to a modular software design.

Nodes are used as an abstraction of an actuator present at the beamline. Each node can request or can be asked about status information about its real-world counterpart. Furthermore, the way services were implemented, nodes can be commanded to move the undulator or monochromator axes, respectively.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP013

Figure 2 shows the simplified operating principle of the implemented ROS2 node network. The node network is running on a RaspberryPi 3B+ which provides two CAN interfaces, used to communicate with the undulator and monochromator motion controllers. The connected motion controllers provide a CANopen API to control the axis movement. Ignoring possible exceptions during the process, performing an energy scan can be described in 5 steps:

1. An outside command to execute a scan in continuous mode is sent to the prototype.
2. The command includes the energy trajectory parameter, like start energy, end energy, $^{\Delta energy}/_{time}$ and $^{\Delta energy}/_{time^2}$. The command also includes the names of the controller nodes which should be involved in the driving process. This way modularity is preserved. Actuators can be excluded from a driving process, or yet to be developed nodes can be integrated in the future. Based on the given energy parameter, the *Scan Generator* node calculates the energy trajectory and exports it as `energy.csv`.
3. The file path to `energy.csv` is transmitted to the controller nodes, defined by the incoming user command from step 1. Based on the energy trajectory, each controller node calculates the movement trajectory for their actuator. It is important that this action is performed by the controller nodes, and not by the Scan Generator node, to achieve modularity. The generated movement trajectories are exported as `trajectory.csv` files.
4. After successfully calculating the movement trajectories, the Scan Generator node commands the controller nodes to move into the start position of the just calculated movement trajectory.
5. In the last step, the system will synchronously start the execution of the speed trajectory for the undulator and monochromator. Regular speed updates are sent to the undulator and monochromator to achieve the calculated movement of both.

The system also implements rudimentary logging capabilities, to be able to compare the requested trajectory and the actual trajectory.

Modularity and Interfaces To achieve modularity a clear interface definition is necessary. For the controller nodes to be interchangeable, each has to provide the identical interface, defined by the following services[1]:

`calc_trajectory_by_energy_csv` : Based on a given energy trajectory, the node will calculate a movement trajectory (see step 3).

`drive_to_start_pos_by_csv_trajectory` : The controller node will position its actuator in the start position, based on the given movement trajectory (see step 4).

`get_driving_status` : Returns the current status of the actuator, the node is controlling. The status contains information about the driving state (see step 4).

[1] Service arguments are omitted for readability reasons

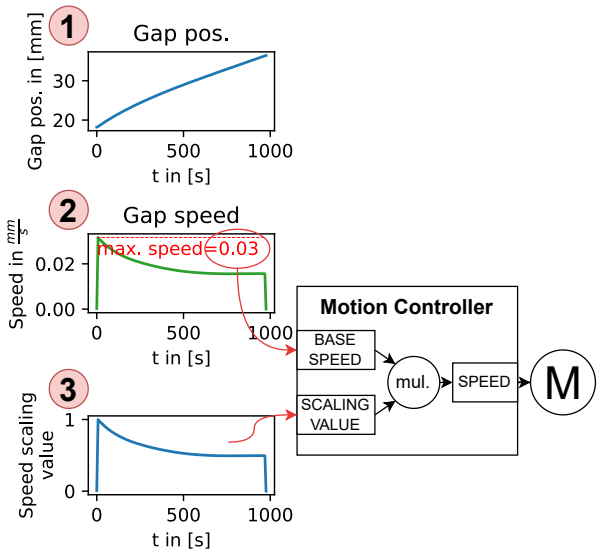

Figure 3: Trajectory transmission process.

`drive_speed_trajectory_by_csv_file` : Commands a controller node, to start driving the given speed trajectory (see step 5).

The previously described scanning process can be extended with arbitrarily many[2] controller nodes, as long as they implement the shown services.

Trajectory Transmission This section describes how the controller nodes (see Fig. 2) transmit the calculated movement trajectory to the undulator and monochromator axes. It describes the process well enough to understand the underlying principle but will not go into details about the CANopen implementation.

Figure 3 shows the basic steps behind the trajectory transmission process. The following steps assume, that the motor is already in the start position.

1. A movement trajectory, based on a given energy range, is generated.
2. By deriving the position trajectory we get the speed trajectory. The maximum speed of the trajectory will be communicated to the motion controller. See `BASE SPEED`.
3. The execution of the trajectory is realized by periodically sending scaling values in regard to the configured `BASE SPEED` to the motion controller (see `SCALING VALUE`). The motion controller then calculates and executes the requested speed based on these two values.

RESULTS AND DISCUSSION

The presented results were recordings of an actual test scan using the UE56/2 undulator, connected to a Unidrive M700 motion controller, and a plane grate monochromator from Jenoptik AG connected to a Beckhoff motion controller CX5130 programmed via TwinCAT 3 NC.

[2] The number of nodes and processes is only limited by the prototype's hardware.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP013

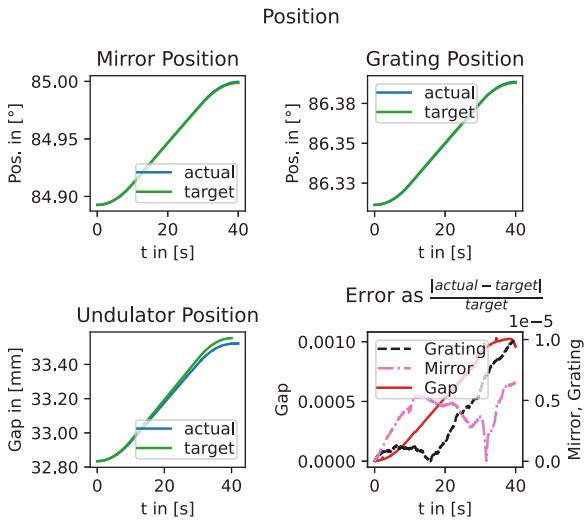

Figure 4: Position plot for an energy range of 700–730 eV with an energy speed of 1 $\frac{eV}{s}$ and an energy acceleration of 0.1 $\frac{eV}{s^2}$.

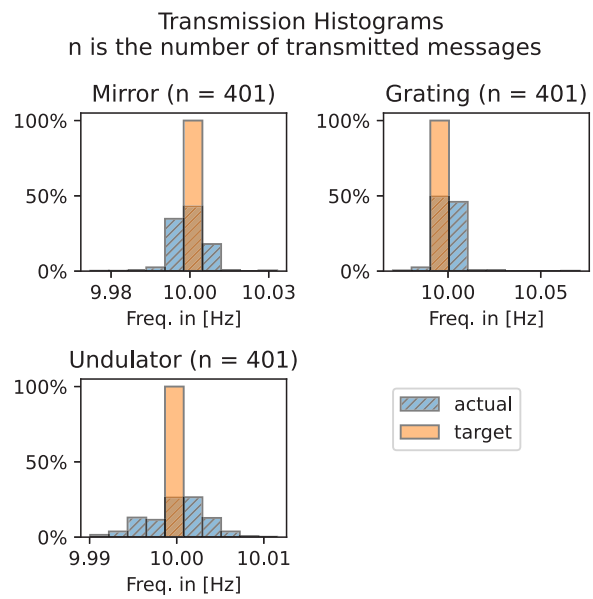

Figure 5: Transmission histogram of an energy range of 700–730 eV with 401 speed updates.

For the undulator motion controller the necessary CANopen API for transmitting the speed trjectory was already implemented. However, the monochromator motion controller required a prototypical implementation of a CANopen API for transmitting the speed trajectory for the mirror and grating movement.

The prototype was tested for scans across multiple energy ranges with different energy speeds and accelerations. Two example trajectories will be discussed.

The discussed main characteristics were found across different test runs using different energy ranges, speed and acceleration. The data was recorded by the developed system itself.

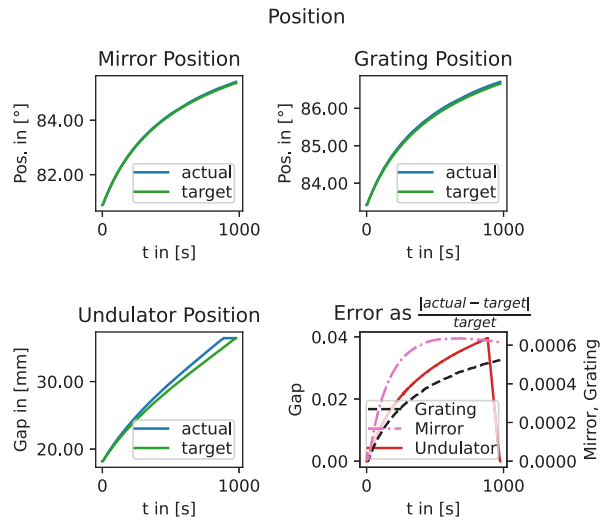

Figure 6: Position plot for an energy range of 220–880 eV with an energy speed of 0.65 $\frac{eV}{s}$ and an energy acceleration of 0.1 $\frac{eV}{s^2}$.

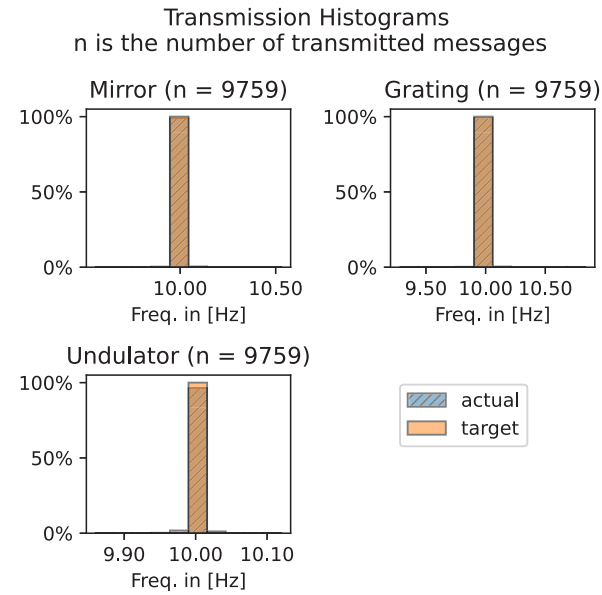

Figure 7: Transmission histogram for an energy range of 220–880 eV with 9759 speed updates.

Figure 4 shows the movement trajectories for a typical scan for iron. This scan is defined by an energy range of 700–730 eV. The chosen energy speed was 1 $\frac{eV}{s}$ with an energy acceleration of 0.1 $\frac{eV}{s^2}$. Therefore, based on the requested energy trajectory, constant rate of energy change would be achieved after 10 s. The plots show that the undulator and monochromator follow the requested trajectory in a general sense.

The data points of the trajectories were transmitted with a frequency of ~10 Hz. Over a trajectory duration of 40 s, 401 speed updates for the monochromator mirror and grating and the undulator gap were sent. Figure 5 shows the transmission histograms for the monochromator grating and mirror as well as for the undulator gap.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP013

To better show the abilities to drive with variable speed, an arbitrary scan with an energy range of 220–850 eV was performed in ~16.3 min. Figure 6 shows the results. The behaviour of the curve represent very well the usage of changing the speed over time.

The data points of the trajectory were transmitted with a frequency of ~10 Hz. For this wide energy range, 9759 speed updates per axis were transmitted over ~976 s. Figure 7 shows the corresponding histograms.

As of the current stage of this project, the positioning error and its scale is expected. To improve the precision of the prototype, extending it into a closed-loop system is necessary.

CONCLUSION AND OUTLOOK

The developed prototype is seen as a success, measured by the self-imposed goals. It has shown that it is possible to implement a system which acts like a coordinator of multiple actuators at BESSY II. Furthermore, the software design, using the ROS2 framework, promises modularity which allows the extension of the system for additional actuators and sensors.

In the current implementation, as an open-loop system, the system does not provide the necessary precision to be used for continuous scans in a production environment. To meet the precision requirements, the system has to be extended to a closed-loop system.

Furthermore, there is the comparatively high cost of implementation compared to using an off-the-shelf motion controller system. Not only the system itself has to be developed, but APIs for the system to connect to have to be developed too. The need for configurability can justify such a time investment.

The cost of development could be reduced by combining efforts of similar projects. One example could be the PandABox. The PandABox utilizes a Xilinx Zynq-7030 SoC to process incoming signals [5]. The user can define trigger signals which, when detected, produces user defined output signals. The ability to read signals, generate signals and to define trigger signals is also a requirement for potential future iterations of the developed prototype. By integrating the ROS2 framework into the PandABox, complex workflows necessary for orchestrating complex beamline architectures would be possible.

ACKNOWLEDGEMENTS

We want to thank Dr. Jens Viefhaus and Olaf-Peter Sauer for their steady encouragement during the development of this project. Special thanks to Edward Rial, Stefan Gottschlich and Stefan Grimmer for providing support during undulator tests and operation. Furthermore, we want to thank Oonagh Mannix for her feedback during the writing of this paper.

REFERENCES

[1] A. F. Balzer, E. Schierle, E. Suljoti, M. Witt, and R. Follath, "Status of the Continuous Mode Scan for Undulator Beamlines at BESSY II", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1091–1095. doi:10.18429/JACoW-ICALEPCS2015-THHA3O02

[2] O. Mathon *et al.*, "The time-resolved and extreme conditions XAS (TEXAS) facility at the European Synchrotron Radiation Facility: the general-purpose EXAFS bending-magnet beamline BM23", *J. Synchrotron Radiat.*, vol. 22, no. 6, pp. 1548–1554, 2015. doi:10.1107/S1600577515017786

[3] A. K. Poswal *et al.*, "Augmentation of the step-by-step Energy-Scanning EXAFS beamline BL-09 to continuous-scan EXAFS mode at INDUS-2 SRS", *J. Synchrotron Radiat.*, vol. 23, no. 6, pp. 1518–1525, 2016. doi:10.1107/S160057751601362X

[4] K. Medjoubi *et al.*, "Development of fast, simultaneous and multi-technique scanning hard X-ray microscopy at Synchrotron Soleil", *J. Synchrotron Radiat.*, vol. 20, no. 2, pp. 293–299, 2013. doi:10.1107/S0909049512052119

[5] S. Zhang *et al.*, "PandABox: A Multipurpose Platform for Multi-technique Scanning and Feedback Applications", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 143–150. doi:10.18429/JACoW-ICALEPCS2017-TUAPL05

[6] T. C. Shen *et al.*, "Exploring Alternatives and Designing the Next Generation of Real-Time Control System for Astronomical Observatories", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 824–828. doi:10.18429/JACoW-ICALEPCS2021-THBL02

[7] B. Kalantari, E. Johansen, *et al.*, "CompactPCI-Serial Hardware Toolbox for SLS 2.0", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper TUAL01, unpublished. https://jacow.org/icalepcs2021/talks/tual01_talk.pdf

[8] Pmod CAN Reference Manual, https://digilent.com/reference/pmod/pmodcan/reference-manual?redirect=1#serial_communication

[9] Linux Kernel Source Code v5.1.18, drivers/net/can/spi/mcp251x.c, https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/net/can/spi/mcp251x.c?h=v5.1.18

[10] Steve Macenski, Alberto Soragna, Michael Carroll, and Zhenpeng Ge, "Impact of ROS 2 Node Composition in Robotic Systems", *arXiv preprint*, p. 09933, May 2023. doi:10.48550/arXiv.2305.09933

[11] Raspberry Pi Logo, Raspberry Pi is a trademark of Raspberry Pi Ltd, https://www.raspberrypi.com/app/uploads/2022/02/COLOUR-Raspberry-Pi-Symbol-Registered.png

[12] ROS Logo, ROS trademarks are property of Open Source Robotics Foundation, Inc., https://fkromer.github.io/awesome-ros2/ros_logo.svg