Software Testing, AI and Robotics (STAIR) Learning Lab

Simon Haller-Seeber [ID], Thomas Gatterer, Patrick Hofmann,
Christopher Kelter, Thomas Auer, and Michael Felderer [ID]

Department of Computer Science, University of Innsbruck
Technikerstr. 21a, 6020 Innsbruck, Austria

Abstract. In this paper we presented the Software Testing, AI and
Robotics (STAIR) Learning Lab. STAIR is an initiative started at the
University of Innsbruck to bring robotics, Artificial Intelligence (AI) and
software testing into schools. In the lab physical and virtual learning
units are developed in parallel and in sync with each other. Its core
learning approach is based the develop of both a physical and simulated
robotics environment. In both environments AI scenarios (like traffic sign
recognition) are deployed and tested. We present and focus on our newly
designed MiniBot that are both built on hardware which was designed for
educational and research purposes as well as the simulation environment.
Additionally, we describe first learning design concepts and a showcase
scenario (i.e., AI-based traffic sign recognition) with different exercises
which can easily be extended.

Keywords: digital twin, educational robotics, physical computing, ar-
tificial intelligence, software testing, internet of things

1 Introduction

We established our school outreach program several years ago, and apart from
some Lego Mindstorms classes we saw very little robotics in Tyrolean schools.
So we started specialized efforts to bring educational robotics and physical com-
puting to schools ([1], [3], [5]). We as well had specialized university courses to
teach our student teachers possibilities on how to do robotics in schools and
guide teachers to build and program robots with their students to participate in
national 'and international robotic competitions.

In general, we want to get young people interested in science, technology,
engineering and mathematics (STEM) matters, and in computer science in par-
ticular. We see physical computing and robotics as an intuitive and motivat-
ing playground – things move and interact, making the everyday relevance of
computer science tangible. The ability to connect physical and virtual objects
(things) with one another in almost any way has grown in importance and now
permeates all areas of life, such as mobility ("smart mobility", "autonomous
driving", "smart city"), housing ("smart home"), agriculture ("smart agricul-
ture"), health ("smart health") or production ("Industry 4.0"). The high practi-
cal importance and the almost unlimited possibilities to realize digital ideas and

arXiv:2204.03028v1 [cs.RO] 6 Apr 2022

products also with AI technologies via e.g. Internet of Things (IoT) systems are in contrast to the high level of technical understanding which is required to grasp these technologies. In order to master the complexity of such systems and to exploit their potential, basic understanding and basic knowledge of the underlying technologies are therefore an indispensable component of an innovation-oriented digital education. The physical computing part additionally can help in forming students' knowledge and thinking and provide better understanding of the effects of mechanics and mathematics via not only pure algorithmic thinking. Problem solving methods are developed, which give pupils and students advantages in later everyday life while laying a foundation for lifelong learning. One of the ways of making digital education more relevant is to integrate such elements of modern technology into appropriate teaching subjects. There are of course initiatives which already implemented such efforts at different educational levels, from school programs to small workshops, but there is very little to none in the Tyrolean area. Additionally, we want to widen our aims compared to our previous initiatives, therefore we further developed our efforts, started a new collaboration with the *Media Inclusion AI Lab* were the outcome is now our joint Learning Space as part of the INNALP Education Hub[1].

The Learning Lab serves the following objectives and goals in the fields of AI, robotics and software testing:

- Activity-based teaching of basic and advanced knowledge in those fields
- Provision and develop learning materials with the involvement of teachers for all ages and for any prior knowledge
- Provide a simulated and physical lab infrastructure to support advanced testing and teaching scenarios

2   Hardware and Software

To give students a good introduction to AI, robotics and software testing, it is important to provide them also with hands-on experience. This can be done in a virtual or a physical environment. To address both options, we do not only look at the hardware capabilities of a physical platform, but also on the available software stack. A physical platform itself should be affordable, easy to use for all ages, robust, and yet be able to showcase research or more complex algorithms. At the moment there is no suitable, off-the-shelf product out there. Either they are really cheap and do not provide any sensory feedback, or they are industrial or research products which are too expensive, the intended usage is too narrow, or they are unreliable. [4], [6]

Therefore a combination of consumer products building up a robot which is capable of performing interesting tasks, ranging from moving around, self-localisation and mapping (SLAM), planning, grasping, object recognition and the ability to use modern state of the art machine learning algorithms, is attractive.

---

[1] https://projekte.ffg.at/projekt/4119035

We conducted a market research and compared them to identify feasible partial solutions to build such a robot.

As a mobile base platform the choice fell on Sphero RVR[2]. It has a broad software support (Python, C++, etc.), the hardware is build in a stable way, includes a vast variety of sensors: a 9-axis IMU (3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer/compass), an rgb color sensor, a distance and light sensor, and it is not too expensive.

For grasping and manipulation we chose the 4 DoF Lynxmotion Arm[3]. The LSS Motors provide additional sensor feedback (voltage, current, temperature, position, and rotation per minute) and they can set properties such as angular stiffness, holding stiffness, acceleration and deceleration. The software support was not too broad but we build upon the complete open reference. As a compute module we chose the Nvidia Jetson Nano[4] in combination with an Intel D435 depth camera[5] which can be used for machine learning algorithms. A picture of the robot and its design is shown in Fig. 1. Because this combination of hard-
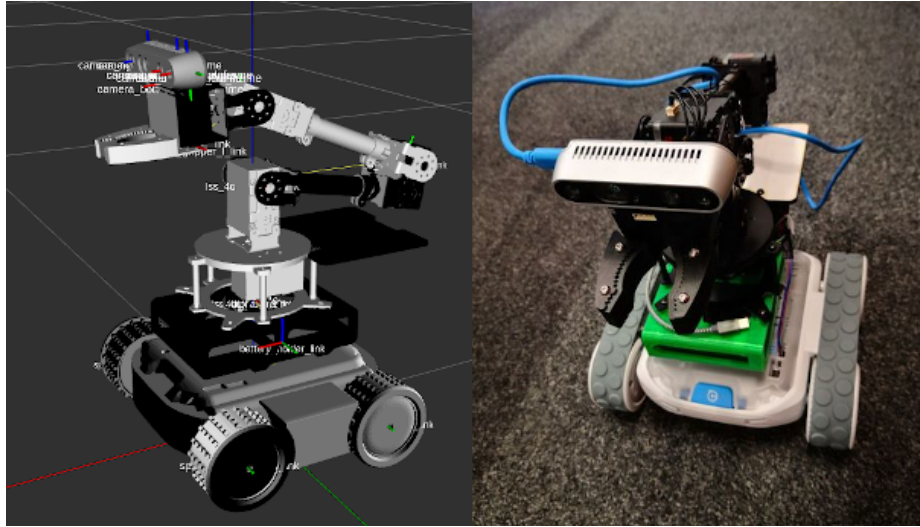

Fig. 1. Sphero RVR with Lynxmotion Arm and in Arm Camera. (Left: in simulation. Right: as physical platform)

ware is new, we provide a standardized way for accessing this robot using a robot operating system (ROS) middelware. We chose ROS in this educational context

---

[2] https://sphero.com/products/rvr
[3] https://robotshop.com/de/de/lynxmotion-lss-4-dof-roboterarm-kit.html
[4] https://developer.nvidia.com/embedded/jetson-nano-developer-kit/
[5] https://www.intelrealsense.com/depth-camera-d435/

because then we have a consistent way to control the simulation and a real robot. Additionally, we can address all age levels from beginners to university students. For beginners we will provide an easy block based web programming interface and a simple python API for intermediate pupils and students, advanced university students can dive into the controller back-end similar to [2]. We provide a full ROS integration for all parts and the complete robot including the following:

Nvidia Jetson: An updated way to build an Ubuntu 20.04 image including ROS Noetic. This is important for the possibility to use python3, ROS and current implementation of machine learning algorithms. We also provide an Package repository for easy OpenCV and Tensorflow installation.

Sphero RVR: We provide a full integration for the base platform, this includes especially: A ROS Description Package (including a URDF model), and a basic ROS node for robot interaction and simulation. Robot control, tf, odometry and other sensor data provisioning is done in such a way that one can benefit from the off-the-shelf ROS ecosystem (see Fig. 2).
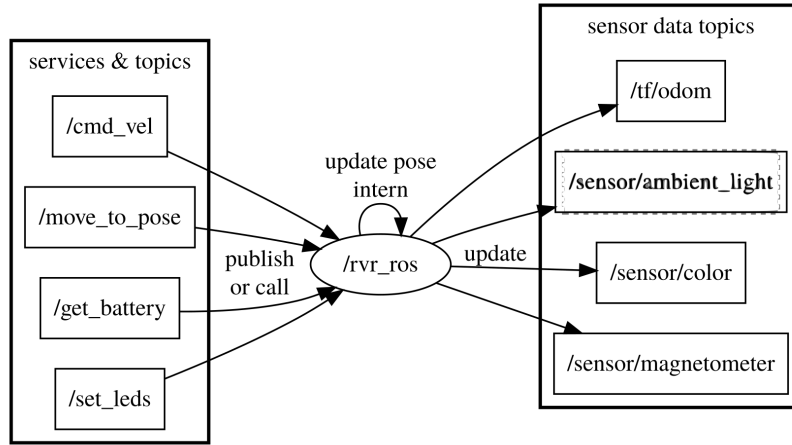

Fig. 2. Sphero RVR ROS Services and Topics

Lynxmotion LSS 4DoF Arm: We provide a full integration for this hardware, which includes: A ROS Description Package (including a URDF model), a basic ROS node for arm interaction and simulation, and a ROS Arm MoveIt package for planning. A simplified view on the current ROS integration of the Arm is shown in Fig. 3.

Intel Camera/AI: Demos and usage examples.

Tips and instructions on how to build the robot hardware, as well as how to install and build the software setup can be found at https://stair-lab.uibk.ac.at/.
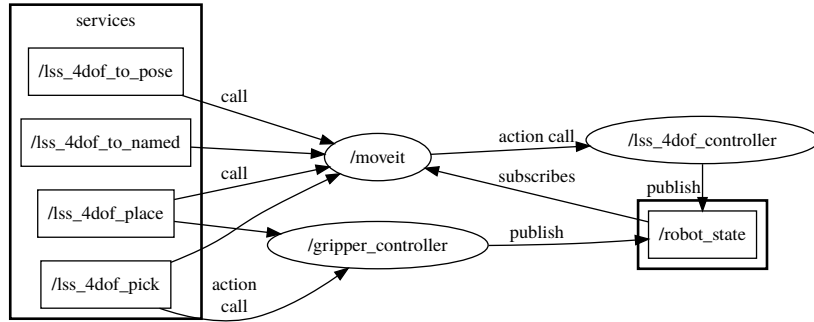
Fig. 3. Simplified view: ROS integration (Services and Topics) for the Arm and Gripper

The code base and implementations for the robot base and arm as well as the build environment for the Nivida Jetson can be found at: https://git.uibk.ac.at/informatik/stair/.

3   The Learning Lab on a Running Example

In our learning environment, we use a digital twin, among other things. This is the virtual representation of our robot and scenario. We use a physical simulation to not only experiment and learn, but also to evaluate exercises. While our digital twin does not control any hardware, it can provide a valuable learning environment outside of our workshops. It offers an easy way to check trained AI models and written software. Additionally, we can do best practice software testing. We run, test and debug all ROS nodes either with a plugin for Visual Studio Code or with a plugin for IntelliJ IDEA.[6] Both work with multiple test frameworks out of the box.

In general one can use our environment with many already existing exercises and tasks from other robots and environments: From easy and beginner level block programming e.g. building a line- or wall-follower, to intermediate ones using our simple python API (which uses our ROS modules): e.g. exploring a maze. In the following section we show-case a scenario which is not very common for beginner level students and briefly present possible extensions for intermediate and advanced learners.

3.1   Traffic Sign Recognition

In the living environment of pupils taking the driver's license test at high-school age, traffic-sign recognition is quite important. They recognize that the car market is changing and that autonomous cars will take over much of the content

---

[6] ROS VS Code Plugin: https://marketplace.visualstudio.com/items?itemName=ms-iot.vscode-ros
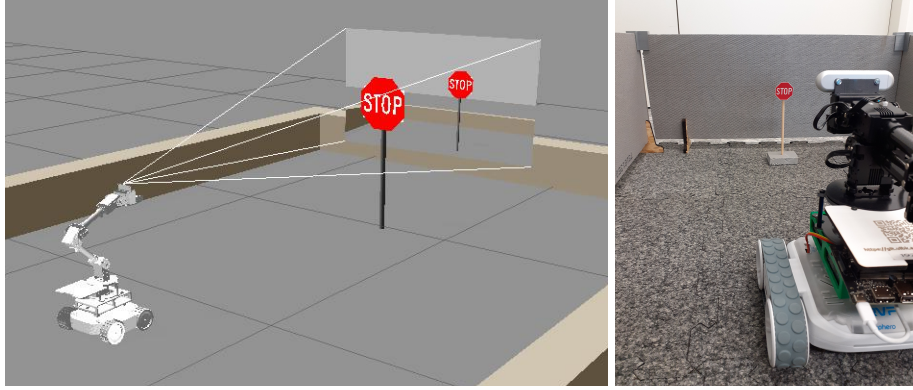IntelliJ Idea Plugin: https://plugins.jetbrains.com/plugin/11235-ros-support

Fig. 4. Our Robot in the traffic sign recognition example scenario (Left: simulated environment; Right: robot in the arena)

they focus on. This gives students a good understanding of what tasks an artificial intelligence has to do and what problems arise. After hands-on training, students can see the robot driving around in a real or simulated world, looking for traffic signs with its camera. Each time a traffic sign is detected, the image processing algorithm sends a command to the robot telling it how to respond to that sign. For example, if the robot finds a "stop" sign, it stops in front of the sign not continuing or going in another direction. A visualisation of the simulated environment and in a real arena is shown in Fig. 4.

In this scenario we have prepared three different exercises for students of different age groups and with different prior knowledge:

Beginner: Train a model with printed and/or self designed traffic signs at Teachable Machine[7]. This model can then be easily incorporated via our ROS Blockly environment (see Fig. 5). Afterwards one has to specify actions for recognized classes in the robot control loop. Because we use ROS it is easy to switch between execution on the real robot and execution on the digital twin.

Intermediate: Train a support vector machine (SVM). Compare the outcome between the beginner and this exercise. This is an interesting task, because one needs just a few training examples and low compute requirements, meaning that this could be done on the Nvidia Jetson Nano.

Advanced: For a more advanced machine learning algorithm exercise we propose to train a YOLO model and test it. The training can't be (realistically) done on the Nvidia Jetson Nano, but it is possible to evaluate this state of the art algorithm on the robot.

---

[7] https://teachablemachine.withgoogle.com/train/image
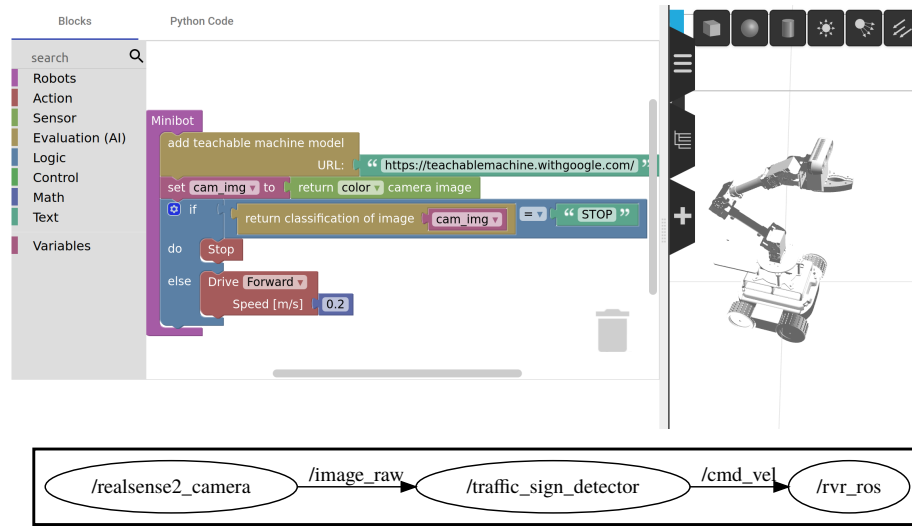
Fig. 5. Left: Blocktype Programming with AI: Example program using a trained model from Teachable Machine; Right: Simulated Robot; Below: Backend - ROS nodes used for this example in our hardware environment

4 Conclusion

In this paper we presented the Software Testing, AI and Robotics (STAIR) Learning Lab. Its core idea is the develop of a physical and simulated robotics environment in parallel and in sync with each other. In both environments AI scenarios (like traffic sign recognition) are deployed and tested.

We managed to build a robot which can be used for various courses in the fields of robotics, AI and software testing. Currently we build one scenario with different exercises for different learning environments. These can be used in simulation environment and on a physical robot. We contributed a complete ROS interface for the Sphero RVR platform, Lynxmotion LSS 4DoF Arm. Additionally, we provide the kinematics of the robot and several examples. As a next development step of our learning lab we will provide easily-accessible courses and workshops around this robot which will then be offered to Tyrolean schools in the upcoming term. In parallel, we will evaluate the workshops and their learning effects to finally provide evidence-based learning units.

Acknowledgement

References

1. Auer, T., Felderer, M.: Towards a learning environment for internet of things testing with lego® mindstorms®. In: 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 457–460 (2020). https://doi.org/10.1109/ICSTW50294.2020.00081
2. Gervais, O., Patrosio, T.: Developing an Introduction to ROS and Gazebo Through the LEGO SPIKE Prime. In: Robotics in Education. pp. 201–209. Springer, Cham (Apr 2021). https://doi.org/10.1007/978-3-030-82544-7_19
3. Haller-Seeber, S., Renaudo, E., Zech, P., Westreicher, F., Walzthöni, M., Vidovic, C., Piater, J.: ROSSINI: RobOt kidS deSIgn thiNkIng. In: Robotics in Education. vol. 1, pp. 16–25. Springer Advances in Intelligent Systems and Computing (01 2021). https://doi.org/10.1007/978-3-030-67411-3_2
4. Karalekas, G., Vologiannidis, S., Kalomiros, J.: Europa: A case study for teaching sensors, data acquisition and robotics via a ros-based educational robot. Sensors **20**(9) (2020). https://doi.org/10.3390/s20092469, https://www.mdpi.com/1424-8220/20/9/2469
5. Lamprecht, P., Haller-Seeber, S., Piater, J.: A Block–based IDE Extension for the ESP32. In: Robotics in Education. vol. 1, pp. 304–310. Springer Advances in Intelligent Systems and Computing (01 2021). https://doi.org/10.1007/978-3-030-67411-3_27
6. Webresource, part of University of Innsbruck course LV703051/19: Mini overview on affordable physical computing platfroms which are suitable to build educational robots (2020), https://docs.google.com/spreadsheets/d/18ftAGsW0e-4IHu9VnFUISlkZ3Pj3ZBPwdiHq7ZgXySw