The Journal of Open Source Software

ARCHES PiCar-X: Software for Digital Twin Research

Alexander Barbie [1¶] **and Wilhelm Hasselbring** [1]

1 Software Engineering Group, Kiel University, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

This paper presents a digital twin prototype of the PiCar-X by Sunfounder based on the middleware Robot Operating System (ROS), Docker, and the ARCHES Digital Twin Framework, which provides tools to exchange data between a physical and digital twin. The digital twin prototype can be used to test all software functions without needing to use the physical PiCar-X. The actual hardware is replaced with emulators or a simulation, and the interfaces are virtualized. Our goal is to provide researchers and practitioners with an affordable and straightforward example to explore how the concepts physical twin, digital model, digital template, digital thread, digital shadow, digital twin, and digital twin prototype can be implemented. These concepts were originally used for development, testing, monitoring, and operating an underwater network of ocean observation systems in the project ARCHES (Autonomous Robotic Networks to help Human Societies).

Statement of need

Digital twins are becoming increasingly relevant in the Industrial Internet of Things and Industry 4.0, enhancing the capabilities and quality of various applications (Kritzinger et al., 2018). However, the concept of digital twins lacks a unified definition and faces validation challenges, partly due to the scarcity of reproducible modules or source codes in existing studies. While many applications are described in case studies, they often lack detailed, re-usable specifications for researchers and engineers.

In (Barbie & Hasselbring, 2024b), we formally specified a digital twin concept including its sub-concepts physical twin, digital model, digital template, digital thread, digital shadow, digital twin, and digital twin prototype using the Object-Z notation. These concepts were developed for a network of ocean observation systems and the results were evaluated in a real-world mission in the Baltic sea in Oktober 2020 (Barbie et al., 2021). One of the results of the successful proof-of-concept was the ARCHES Digital Twin Framework (Barbie & Pech, 2022), a software package providing the functionality to implement the digital thread between physical twins and digital twins. Ocean observation systems use quite specific and expensive hardware, hence, we see the need of a cheap lab experiemnt to enable independent evaluation and exploration of the different concepts. The PiCar-X example demonstrates all the concepts from the ocean observation system and includes also a full integration test pipeline, see our GitHub Repository (Barbie & Hasselbring, 2024a). In (Barbie & Hasselbring, 2024c), we elaborate in more detail how the PiCar-X can be used to evaluate all these concepts.

The PiCar-X and its Digital Twin

There is no clear definition of a digital twin. The range of interpretations spans from a complex simulation to a completely mirrored status of the physical device, which can also be controlled via the digital twin. In our definition (Barbie & Hasselbring, 2024b), the digital twin is connected to the physical twin over the entire life cycle for automated bidirectional data exchange, i.e. changes made to the digital twin lead to adapted behavior of the physical twin and vice-versa. The implementation of this capabilties can very, with model driven approachen are being very popular (Barbie & Hasselbring, 2024b). Hence, we present a lab experiment that can be used to explore our approach independely.

The PiCar-X is a toy car, see Figure 1, with all sensors and actuators connected to a RaspberryPi. Two direct current motors (DC motors) are used to move the car. A servo motor at the front is used to steer the car. The steering is a typical Ackermann steering (Veneri & Massaro, 2020) known from common cars. The PiCar-X also includes grey-scale sensors for line following, infrared sensors for collision avoidance, and a camera. However, in the current example only the DC motors and the servo motor for steering are included, so far. All software components are implemented using the middleware ROS and are containerized using Docker. The ARCHES Digital Twin Framework establishes the digital thread between the physical twin and the digital shadow/twin (Barbie & Hasselbring, 2024c).

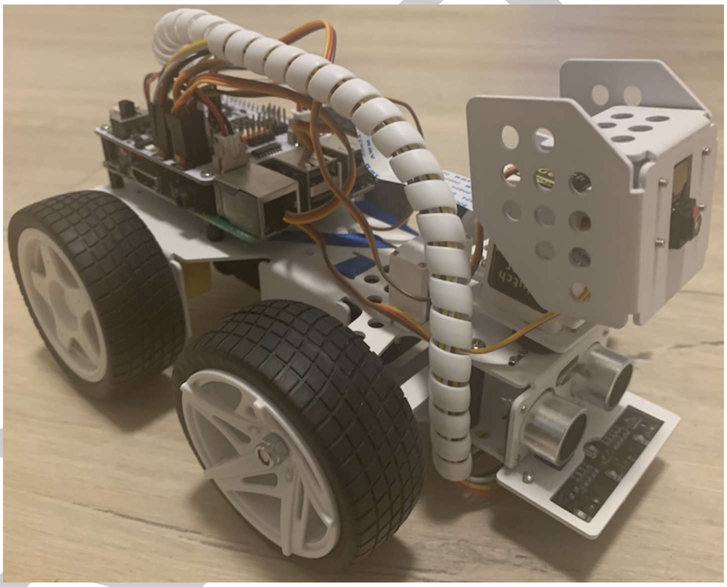

Figure 1: The Picar-X by SunFounder.

Lacking official CAD files for the PiCar-X, we utilized a simplified CAD model of an older SunFounder PiCar version, see Figure 2, available under Apache 2.0 license on GitHub. This model, consisting of just the frame and wheels, closely mirrors the original PiCar-X's key dimensions like wheelbase and track, crucial for an accurate Ackermann steering simulation. However, the original PiCar-X's steering mechanism, operated by a steering bar to achieve Ackermann angles, could not be replicated in GAZEBO. Instead, we approximate the Ackermann steering angles for both front wheels based on established methodologies (Veneri & Massaro, 2020).

The Journal of Open Source Software

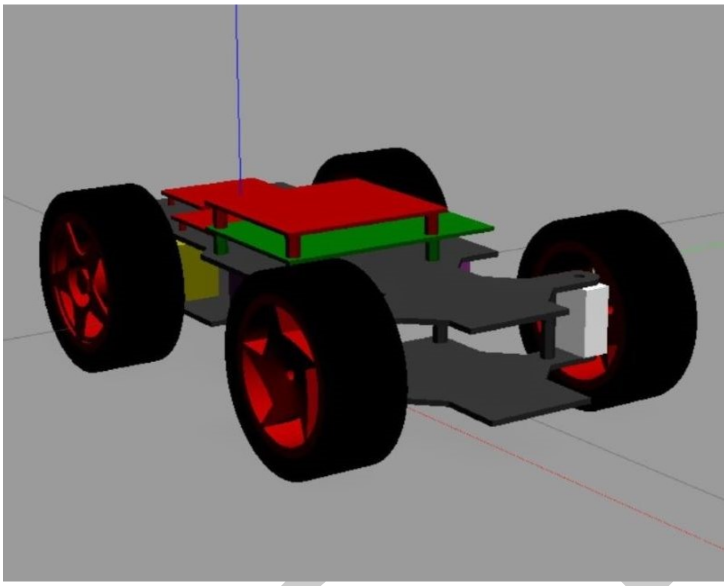

Figure 2: The CAD model used for the PiCar-X digital model in a GAZEBO simulation.

In this project, we provide Docker compose files that can be used to run the software with either the digital model, digital shadow, or digital twin. Our approach to implementing a digital twin differs from others by reusing as many software components from the physical twin as possible for the digital twin (Barbie & Hasselbring, 2024c).

The Digital Twin Prototype

Developing a physical twin typically requires connecting the hardware to a development environment. However, in this setup, only one person can use the hardware at a time. For a team of engineers, this means either everyone needs their own PiCar-X or they must take turns, which can become costly, especially in real-world applications like full-scale vehicles. This is why we developed the digital twin prototype. A digital twin prototype serves as the software counterpart of a physical twin, with identical configurations (Barbie & Hasselbring, 2024b). However, instead of physical sensors and actuators, emulators are used to mimic their functions. These emulators utilize existing sensor and actuator recordings to emulate the physical twin's behavior. This allows the digital twin prototype to replace the physical twin during development and for integration testing in CI/CD pipelines.

The core of the digital twin prototype approach involves replacing all physical sensors and actuators with emulations or simulations, effectively virtualizing the hardware interfaces. As a result, the device driver cannot - and does not need to - distinguish between a real sensor/actuator and its emulated equivalent. The ARCHES PiCar-X uses emulators connected to a GAZEBO simulation. The simulation provides the virtual context for the emulators, instead of using recordings from previous runs.

For the PiCar, the primary interfaces, GPIO and I2C, are emulated using Linux kernel tools. The virtual GPIO interaction module (gpio-mockup) and the I2C chip (I2C-stub) are integrated into the container for these emulation purposes. This example also works on Windows with the Linux subsystem (WSL2). This setup provides a flexible and adaptable environment for emulating the PiCar's hardware interactions. The configuration of the digital twin prototype is illustrated in Figure 3. The digital twin prototype can also be started using the provided Docker compose files.

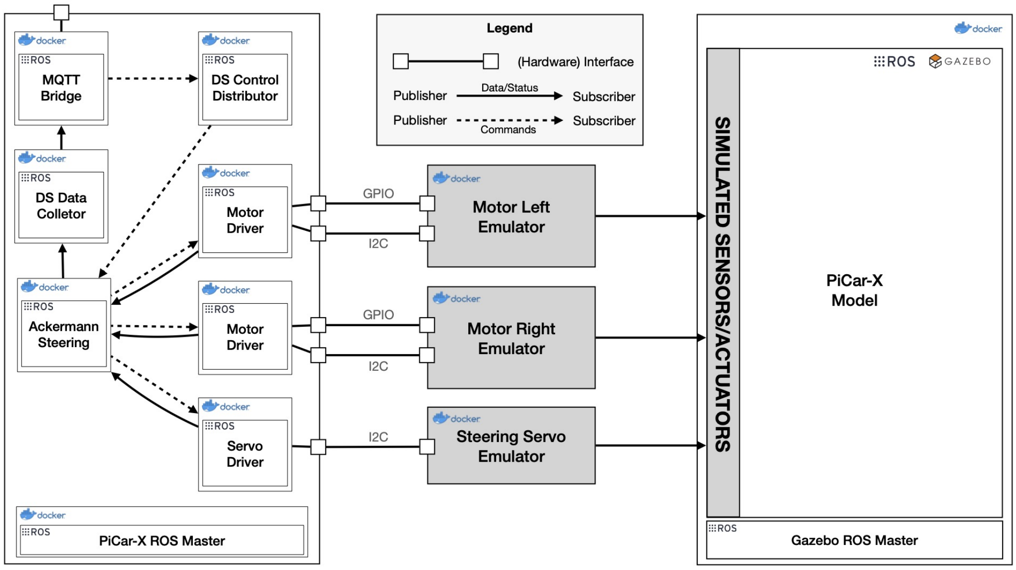

Figure 3: The digital twin prototype of the PiCar-X.

Automated Integration Testing of Embedded Software in CI/CD Pipelines

Test automation has been identified as one of the most prominent areas in the testing of embedded software (Garousi et al., 2018). However, achieving effective automated quality assurance remains challenging, mainly due to the need for hardware involvement in the testing process. Testing on the actual system often requires a continuous connection to the testing environment, which can be expensive and impractical, especially for small and medium-sized enterprises (Barbie et al., 2024). To address this issue, the digital twin prototype is designed to operate independently of the physical system while still enabling the testing of real embedded software.

Figure 4 illustrates an automated CI/CD pipeline for the ARCHES PiCar-X. Whenever a user commits changes, a GitHub Runner is triggered to build a Docker container. This container loads all dependencies and compiles the code. The build step could also include static software checks to further evaluate the code's quality. Once the containers are successfully built, unit tests are run on the module under test. If these tests pass, the process moves to the next crucial phase: integration testing.

To demonstrate how the digital twin prototype can be leveraged for automated integration testing without requiring physical hardware, we created an integration test based on the script used for speed measurement (Barbie & Hasselbring, 2024c). This test ensures that the digital model in the simulation operates at the same speed as the real one. After passing the integration tests, the various Docker containers are released.

The CI/CD pipelines are executed on three runners in parallel: one for x64 systems, one on a Raspberry Pi 3 (arm32v7), and another on a Raspberry Pi 4 (arm64v8). Note that only the x64 runner can execute the integration tests using the virtual context from the GAZEBO simulation, as GAZEBO does not have an ARM build available.

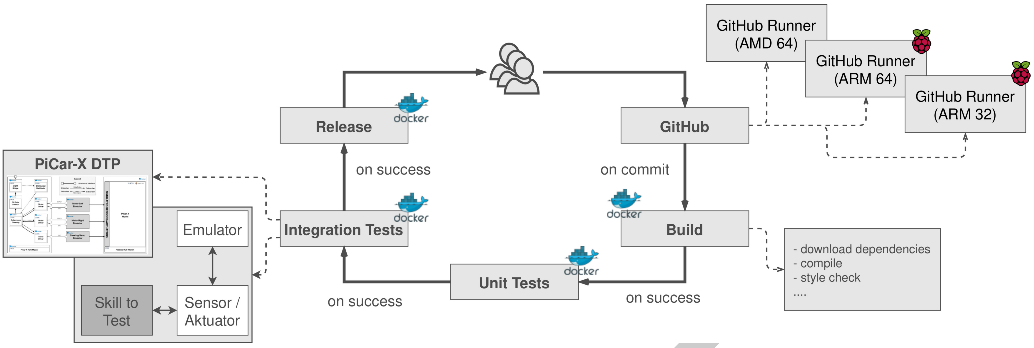

Figure 4: CI/CD Pipeline for the PiCar-X.

Acknowledgements

We thank the GitHub user Theosakamg for providing a CAD model of a PiCar-V under an open source license.

References

Barbie, A., & Hasselbring, W. (2024a). ARCHES PiCar-X. In *GitHub repository*. GitHub. https://github.com/cau-se/ARCHES-PiCar-X

Barbie, A., & Hasselbring, W. (2024b). From Digital Twins to Digital Twin Prototypes: Concepts, Formalization, and Applications. *IEEE Access*, *12*, 75337–75365. https://doi.org/10.1109/access.2024.3406510

Barbie, A., & Hasselbring, W. (2024c). *Toward Reproducibility of Digital Twin Research: Exemplified with the PiCar-X*. https://doi.org/10.48550/ARXIV.2408.13866

Barbie, A., Hasselbring, W., & Hansen, M. (2024). Digital Twin Prototypes for Supporting Automated Integration Testing of Smart Farming Applications. *Symmetry*, *16*(2), 221. https://doi.org/10.3390/sym16020221

Barbie, A., & Pech, N. (2022). *ARCHES Digital Twin Framework*. GEOMAR Helmholtz Centre for Ocean Research Kiel. https://doi.org/10.3289/sw_arches_core_1.0.0

Barbie, A., Pech, N., Hasselbring, W., Flogel, S., Wenzhofer, F., Walter, M., Shchekinova, E., Busse, M., Turk, M., Hofbauer, M., & Sommer, S. (2021). Developing an Underwater Network of Ocean Observation Systems with Digital Twin Prototypes - A Field Report from the Baltic Sea. *IEEE Internet Computing*. https://doi.org/10.1109/mic.2021.3065245

Garousi, V., Felderer, M., Karapıçak, Ç. M., & Yılmaz, U. (2018). What We Know about Testing Embedded Software. *IEEE Software*, *35*(4), 62–69. https://doi.org/10.1109/MS.2018.2801541

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Veneri, M., & Massaro, M. (2020). The effect of Ackermann steering on the performance of race cars. *Vehicle System Dynamics*, *59*(6), 907–927. https://doi.org/10.1080/00423114.2020.1730917