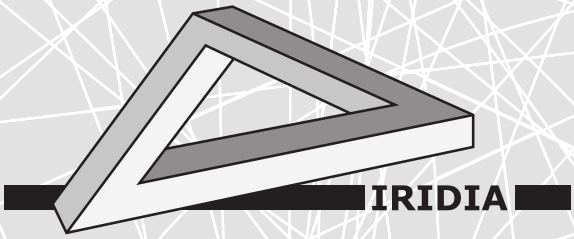

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

DeimOS: a ROS-ready operating system for the e-puck

M. Kegeleirs, D. Garzón Ramos, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2022-013

November 2022

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2022-013

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

DeimOS: a ROS-ready operating system for the e-puck*

Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari
IRIDIA, Université libre de Bruxelles, Belgium

Abstract

Robot Operating System (ROS) is a set of development tools that has become a *de facto* standard both in the academia and the industry. However, little effort has been devoted to enable ROS in small robots widely used in swarm robotics for education and research, such as the e-puck. We developed `DeimOS`: a Linux-based ROS-ready operating system for the e-puck robot extended with the Gumstix Overo COM. `DeimOS` integrates support for ARGoS3—a swarm-dedicated simulator that facilitates the realization of control software for robot swarms. Thanks to `DeimOS`, e-puck robots can be modernized to conduct educational and research experiments in nowadays engaging robotics tasks such as robot mapping. `DeimOS` is a free and open-source software-only solution to extend the lifetime and the capabilities of e-puck robots—a cost effective alternative to existing hardware-based solutions.

Keywords: e-puck; ROS; swarm robotics; Gumstix.

1 Overview

1.1 Motivation and significance

Robots are becoming ubiquitous in our everyday lives. News are full of robots that conquer new grounds (e.g., new operational environments) and that enter into society with novel applications [1]. Robotics education runs alongside this societal trend [2]. Departing from the typical case of programming a single robot, there is a growing interest in studying the realization of groups of robots that can cooperate and act collectively [1, 3]. Swarm robotics [4] is an approach to design collective behaviors for such groups of robots. Education in swarm robotics is a valuable tool to foster learning of transverse skills in interdisciplinary engineering education by tackling challenges in robotics cooperation [5, 2].
A robot swarm is a group of relatively simple robots whose strength builds upon the large number of individuals in the group. Typically, the robots in a swarm are simple in design, hardware and software—making them easy-to-produce and cost-effective devices [6]. This is the case of the e-puck [7]: a generic small mobile robot designed for education. Originally introduced in 2009, the e-puck is probably the most used robot in swarm robotics education and research [8]. The basic form of the e-puck is a two-wheeled robot endowed with infrared proximity sensors and a low-performance microcontroller. Beyond its basic form, the e-puck can be enhanced with new sensors, actuators, and computational power through plug-in extensions. Although the e-puck has been extensively used in the past, no common framework or set of software tools exists to experiment with it. In most cases, users develop *ad hoc* control software and interfaces to interact with the robot—which are typically not reusable. Indeed, nowadays common mobile robot functionalities like autonomous navigation and obstacle avoidance need to be implemented every time from the ground up when experimenting with the e-puck.

The Robot Operating System (ROS) [9] is a popular suit of robotics development tools that can serve as a standard framework to develop control software for robots. ROS provides tools and software libraries to develop applications that cover the design, implementation, and operation of robots—most of them being the standard in the academia and the industry. A large and growing community supports

*The software was implemented by MK. The paper was drafted MK and DGR and edited by MB; all authors read and commented the final version. The research was directed by MB.

ROS by developing reusable software tools that are shared in open-source repositories. We believe that enabling ROS for the e-puck will reduce the burden on educators and researchers in implementing basic robot functionalities. This will concentrate the focus on challenges that are typical in the operation and coordination of multiple robots [10, 11]. However, enabling ROS for the e-puck requires to meet the minimal computing requirements to install it and to provide a basic set of software tools to bootstrap the development of control software for the robot.

At the moment, two commercial computing modules are available for extending the computational power of the e-puck: the Gumstix Overo COM extension board[1] and the Pi-puck extension board [12, 13]. Alongside the commercial modules, researchers have also introduced an *ad hoc* computing module that uses Hardkernel Odroid XU4 [14]—authors coined the name Xpuck for e-pucks using this module. The Gumstix Overo COM has been largely used with the e-puck since its release in 2010, and the latter is a newer module commercially introduced in 2019. E-pucks that use the Pi-puck extension board, as well as the Xpucks, have ROS-ready operating systems: ROS Melodic for the former, and ROS Indigo for the latter. However, ROS is unavailable for e-pucks equipped with the Gumstix Overo COM extension board and its companion Omnivision Module extension—to which we refer as *G-pucks* in the rest of the paper. G-pucks have been traditionally operated through the Ångström[2] operating system: an old and deprecated Linux distribution that does not support ROS. At the moment, the only option available to G-pucks users for enabling ROS is to upgrade the robots by acquiring Pi-puck extension boards. Acquiring new hardware modules might quickly turn in a large economic investment if one considers the large number of robots that are typically used in swarm robotics education and research. Moreover, the recent global shortage of electronics material and devices, in particular Raspberry Pi, has shown that such upgrade are also highly sensitive to external and unpredictable issues.

We present `DeimOS`, an open-source Linux-based operating system for the G-puck providing support for ROS Melodic. `DeimOS` is a software-only solution to enable ROS for G-pucks. We developed `DeimOS` under the framework of the Yocto Project[3] and we configured it to operate with the core ROS software packages. Thanks to the modular nature of the Yocto Project, `DeimOS` allows one to easily add, remove, share, and reuse ROS packages that can bootstrap the development of new software for the robot. `DeimOS` integrates support for ARGoS [15]—a swarm dedicated simulator widely used in education and research. Users of G-pucks can design and test their control software in simulation using the tools provided by ARGoS and later port it to the robots. G-pucks that use `DeimOS` can be integrated in ROS-based applications altogether with e-pucks endowed with other extension modules such as the Pi-puck extension board or *ad hoc* implementations like the Xpuck. `DeimOS` hence offers an alternative to modernize the e-puck with ROS so that it remains a suitable platform to conduct nowadays common robotics experiments in education and research.

1.2 Impact

It is our contention that `DeimOS` provides an easy-to-use solution for teaching (swarm) robotics with the G-puck. Teaching robotics to new generations has recently received growing attention [2]. Indeed, students interested in robotics are frequently repelled by the many, multi-faceted challenges of this field, especially concerning the hardware. Providing robust, well documented platforms is therefore mandatory to promote a progressive learning of the complexity of robotics.

`DeimOS` can also strongly benefit the research community in swarm robotics as nowadays there is still a large body of literature that reports experiments with G-pucks [16, 17, 18, 19, 20]. The development and deployment of software in a ROS-ready system facilitates the initialization, verification and evaluation of the experiments. ROS provides tools that are useful for monitoring and managing experiments such as *rosbag*[4]—a logging and replay tool, and *rviz*[5]—a monitoring and control interface with plugins for a wide variety of sensors and actuators. All desirable tools in the experimentation with large groups of robots.

`DeimOS` enables an easy and convenient way of performing heterogeneous swarm experiments by including both G-pucks and e-pucks using the Pi-puck extension board through ROS communication. Yet, we expect that DeimOS can be easily implemented in new platforms. Indeed, the Yocto Project—

[1] https://www.gctronic.com/doc/index.php/Overo_Extension
[2] http://www.angstrom-distribution.org/
[3] https://www.yoctoproject.org/
[4] http://wiki.ros.org/rosbag
[5] http://wiki.ros.org/rviz

on which we base the development of DeimOS—provides the necessary tools to port our configuration to other robot platforms. We include configuration files to compile `DeimOS` for systems that use other versions of Gumstix—such as the DuoVero and Verdex Pro series.

We have mainly used DeimOS in the context of the emergent research on swarm SLAM [21, 22, 23]. In the current release of DeimOS, we provide core functionalities of ROS and the software packages required to experiment with swarm SLAM. We expect that DeimOS can also contribute to research in human-swarm interaction [24], as ROS provides useful tools to monitor and interact with the robots at runtime. By extending the capabilities of the G-puck, we believe that `DeimOS` will benefit our current research on the automatic design of robot swarms [25, 26]—in which the complexity of the tasks that can be performed by swarms of e-pucks has been limited by the few capabilities available to the robots [27, 17].

1.3 Conclusions

We presented `DeimOS`, a Linux-based ROS-ready operating system for the G-puck robot. `DeimOS` provides tools for using modern robotics software on the e-puck. By integrating ROS alongside libraries from ARGoS3, `DeimOS` leverages a large set of new and useful tools for both education and research—for example, runtime monitoring of robots, and logging and replay of experiments. `DeimOS` extends the number of feasible tasks that can be performed the e-puck. A proof of this potential is the recent experiments on swarm SLAM—a task whose complexity is beyond what could be possibly achieved with the original operating system of the robot. We believe that `DeimOS` supports further complex experimentation with the e-puck, and additional packages can be included thanks to the modularity provided by the framework of the Yocto Project.

2 Implementation and architecture

2.1 Software Architecture

`DeimOS` is based on the Yocto Project, whose distributions are composed of layers that regroup packages and configuration files according to the targeted platform architecture. The definition and configuration of each package in a layer is determined by specific files named *recipes*. Specifically, `DeimOS` is based on the Warrior branch of the Gumstix Repo Manifest for the Yocto Project Build System[6], which adds a Gumstix-specific layer to the Yocto Project Poky standard distribution. Although Warrior is not the latest available Yocto distribution, the Overo Gumstix of the G-puck seems not to support later releases such as Dunfell. `DeimOS` includes the latest ROS distribution available for Warrior, ROS Melodic. We included the recipes for the core functionalities of ROS, and additional ROS packages that are being used in swarm robotics experiments: GMapping [28, 29], multirobot_map_merge [30] and multimaster_fkie [31]. In addition, we integrated an e-puck layer that provides recipes for ARGoS3 along with a template recipe useful to develop new control software for the e-puck with ARGOS3 and ROS—this template handles the initialization and operation of hardware components of the robot. The templates we provide allow developers to easily conceive new control software for the robot, and also, to integrate any software package of the rich ROS ecosystem. The installation instructions can be found at `https://github.com/demiurge-project/DeimOS`.

2.2 Software Functionalities

We present here the main software packages of the distribution.

2.2.1 Robot Operating System software

On top of the core ROS packages that enable the ROS software management and logging, `DeimOS` includes additional specific packages for robot mapping and communication:

(i) *GMapping*[7] is a single-robot, self localization and mapping (SLAM) algorithm that takes sensor information and produces a two-dimensional occupancy grid of the environment. Robots can use this package to produce a map that describes the empty and obstructed areas that the robot encounters.

[6] `https://github.com/gumstix/yocto-manifest`
[7] `http://wiki.ros.org/gmapping`

(ii) *multirobot_map_merge*[8] is a map merging algorithm that enables the fusion of an arbitrary number of maps at run time—in swarm robotics experiments, it is indeed useful to merge the different maps built by the large number of robots of the swarm.

(iii) *multimaster_fkie*[9] allows multiple systems using ROS to communicate. It provides management of the information flow, and embeds discovery and synchronization processes that allow new systems to be added or discarded on the fly. This property is specially relevant in swarm robotics to maintain the scalability of the system.

2.2.2 ARGoS3 Software

ARGoS3 [15] is a swarm-dedicated simulator that allows one to use the same control software for both simulated and real robots. The version of the ARGoS control software included in `DeimOS` is argos3-beta48 (which is the latest version supported by the argos3-epuck plugin, see below). ARGoS3 supports natively the basic e-puck platform, but not its hardware extensions. However, it can be easily extended through plugins to support them. In particular, the argos3-epuck plugin [32] provides support for configurations of the e-puck[10] extended with the range & bearing board (e-RandB), ground sensors, and camera (Omnivision module).

2.2.3 Templates

We provide two Yocto recipe templates that serve as basis to develop new software packages. One template allows one to easily conceive ROS-ready and ARGoS3 control software, and a second one is intended to enable the development of custom ROS-only software.

3 Quality control

We used `DeimOS` in swarm robotics experiments that require the software packages provided by ROS. In particular, we have recently focused on conducting swarm SLAM experiments [21]. In Kegeleirs et al [22], we used a swarm of 10 e-puck robots to map different bounded indoor environments. In our research, we developed control software for several exploration strategies reported in the literature, and evaluated the performance of these strategies on swarm SLAM. First, individual maps were produced by each e-puck using GMapping, and later, we merged the maps using the multirobot_map_merge ROS package. Additionally, we conducted related experiments. For example, we merged maps of two e-pucks directly on the robots thanks to the inter-robot communication capabilities enabled by multimaster_fkie.

Finally, `DeimOS` allowed us to remotely monitor and control e-puck robots. We used the templates to produce control software that integrates ARGoS3 and ROS. The control software enabled the e-pucks to deliver sensor information at runtime, and to receive control commands from an external operator. We used the tools provided by ROS to interface and operate the e-pucks with a Sony PlayStation Dualshock 4 controller[11]. None of the aforementioned experiments would have been feasible with the G-puck without the integration of ARGoS3 and ROS that `DeimOS` provides.

4 Availability

4.1 Operating system

DeimOS can be compiled in standard Linux operating systems (tested on Ubuntu 16.04 and newer).

4.2 Programming language

Yocto Project Warrior (2.7.4).

[8] http://wiki.ros.org/multirobot_map_merge
[9] http://wiki.ros.org/multimaster_fkie
[10] http://www.e-puck.org
[11] https://www.playstation.com/en-us/explore/accessories/gaming-controllers/dualshock-4/

4.3 Additional system requirements

No special requirements are required to compile DeimOS. The operating system can be compiled in typical desktop machines produced in the last five years. DeimOS will run on e-puck robots embedded with Gumstix extension boards.

4.4 Dependencies

No additional dependencies are required.

4.5 Software location:

Code repository,

Name: GitHub

Persistent identifier: https://github.com/demiurge-project/DeimOS

Licence: GPL

Date published: 28/04/2021

4.6 Language

English

5 Reuse potential

We have outlined the reuse potential of `DeimOs` in the above discussion. By integrating ROS alongside libraries from ARGoS3, `DeimOS` provides a new set of tools that extends the number of feasible tasks that can be performed with swarm-dedicated robots like the e-puck. Thanks to the modularity provided by the framework of the Yocto Project, `DeimOS` can be customized to fit the requirements of specific experiments. In particular, we believe that it could enable the interoperability of multiple types of e-pucks in the context of heterogeneous swarm robotics experiments.

Funding statement

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681872) and from Belgium's Wallonia-Brussels Federation through the ARC Advanced Project GbO–Guaranteed by Optimization. MK and MB acknowledge support from the Belgian *Fonds de la Recherche Scientifique*—FNRS. DGR acknowledges support from the Colombian Ministry of Science, Technology and Innovation—Minciencias.

References

[1] Marco Dorigo, Guy Theraulaz, and Vito Trianni. Reflections on the future of swarm robotics. *Science Robotics*, 5:eabe4385, 2020.

[2] Heiko Hamann, Carlo Pinciroli, and Sebastian Von Mammen. A gamification concept for teaching swarm robotics. In Jan Haase, editor, *2018 12th European Workshop on Microelectronics Education (EWME)*, pages 83–88, Piscataway, NJ, USA, 2018. IEEE.

[3] Heiko Hamann. *Swarm robotics: a formal approach*. Springer, Cham, Switzerland, 2018.

[4] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.

[5] Alessandra Vitanza, Paolo Rossetti, Francesco Mondada, and Vito Trianni. Robot swarms as an educational tool: the Thymio's way. *International Journal of Advanced Robotic Systems*, pages 1–13, 2019.

[6] Lorenzo Garattoni and Mauro Birattari. Swarm robotics. In John G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 1–19. John Wiley & Sons, Hoboken, NJ, USA, 2016.

[7] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alchero Martinoli. The e-puck, a robot designed for education in engineering. In Paulo Gonçalves, Paulo Torres, and Carlos Alves, editors, *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, Castelo Branco, Portugal, 2009. Instituto Politécnico de Castelo Branco.

[8] Marco Dorigo, Guy Theraulaz, and Vito Trianni. Swarm robotics: past, present, and future [point of view]. *Proceedings of the IEEE*, 109(7):1152–1165, 2021.

[9] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In Kinugawa Kosuge, editor, *ICRA workshop on open source software*, volume 3, page 5, Piscataway, NJ, USA, 2009. IEEE.

[10] Juan Jesús Roldán, Elena Peña Tapia, David Garzón Ramos, Jorge de León, Mario Garzón, Jaime del Cerro, and Antonio Barrientos. Multi-robot systems, virtual reality and ROS: developing a new generation of operator interfaces. In Anis Koubâa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 3)*, volume 778 of *Studies in Computational Intelligence*, pages 29–64. Springer, Cham, Switzerland, 2019.

[11] Mario Garzón, João Valente, Juan Jesús Roldán, David Garzón Ramos, Jorge de León, Antonio Barrientos, and Jaime del Cerro. Using ROS in multi-robot systems: Experiences and lessons learned from real-world field tests. In Anis Koubâa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 2)*, volume 707 of *Studies in Computational Intelligence*, pages 449–483. Springer, Cham, Switzerland, 2017.

[12] Alan G Millard, Russell Joyce, James A Hilder, Cristian Fleşeriu, Leonard Newbrook, Wei Li, Liam J McDaid, and David M Halliday. The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 741–748. IEEE, 2017.

[13] Jacob M Allen, Russell Joyce, Alan G Millard, and Ian Gray. The Pi-puck ecosystem: hardware and software support for the e-puck and e-puck2. In *International Conference on Swarm Intelligence*, pages 243–255. Springer, 2020.

[14] Simon Jones, Matthew Studley, Sabine Hauert, and Alan Frank Thomas Winfield. A two teraflop swarm. *Frontiers in Robotics and AI*, 5:11, 2018.

[15] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[16] Ken Hasselmann, Antoine Ligot, Julian Ruddick, and Mauro Birattari. Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms. *Nature Communications*, 12:4345, 2021.

[17] David Garzón Ramos and Mauro Birattari. Automatic design of collective behaviors for robots that can display and perceive colors. *Applied Sciences*, 10(13):4654, 2020.

[18] Nadia Nedjah and Luneque Silva Junior. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm and Evolutionary Computation*, 50:100565, 2019.

[19] Muhammad Salman, Antoine Ligot, and Mauro Birattari. Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Computer Science*, 5:e221, 2019.

[20] Lorenzo Garattoni and Mauro Birattari. Autonomous task sequencing in a robot swarm. *Science Robotics*, 3(20):eaat0430, 2018.

[21] Miquel Kegeleirs, Giorgio Grisetti, and Mauro Birattari. Swarm SLAM: challenges and perspectives. *Frontiers in Robotics and AI*, 8:23, 2021.

[22] Miquel Kegeleirs, D. Garzón Ramos, and M. Birattari. Random walk exploration for swarm mapping. In Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang, editors, *Towards Autonomous Robotic Systems, TAROS*, volume 11650 of *LNCS*, pages 211–222, Cham, Switzerland, 2019. Springer.

[23] Ragesh K Ramachandran, Zahi Kakish, and Spring Berman. Information correlated Lévy walk exploration and distributed mapping using a swarm of robots. *IEEE Transactions on Robotics*, 36(5):1422–1441, 2020.

[24] Andreas Kolling, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. Human interaction with robot swarms: a survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2016.

[25] Mauro Birattari, Antoine Ligot, Darko Bozhinoski, Manuele Brambilla, Gianpiero Francesca, Lorenzo Garattoni, David Garzón Ramos, Ken Hasselman, Miquel Kegeleirs, Jonas Kuckling, Federico Pagnozzi, Andrea Roli, Muhammad Salman, and Thomas Stützle. Automatic off-line design of robot swarms: a manifesto. *Frontiers in Robotics and AI*, 6:59, 2019.

[26] Mauro Birattari, Antoine Ligot, and Ken Hasselmann. Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms. *Nature Machine Intelligence*, 2(9):494–499, 2020.

[27] Gianpiero Francesca and Mauro Birattari. Automatic design of robot swarms: achievements and challenges. *Frontiers in Robotics and AI*, 3(29):1–9, 2016.

[28] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In Alícia Casals, editor, *IEEE International Conference on Robotics and Automation, ICRA*, pages 2432–2437, Piscataway, NJ, USA, 2005. IEEE.

[29] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[30] Jiří Hörner. Map-merging for multi-robot system. Master's thesis, Univerzita Karlova, Prague, Czech Republic, 2016.

[31] Sergi Hernández Juan and Fernando Herrero Cotarelo. Multi-master ROS systems. Technical Report IRI-TR-15-1, IRI, Spain, 2015.

[32] Lorenzo Garattoni, Gianpiero Francesca, Arne Brutschy, Carlo Pinciroli, and Mauro Birattari. Software infrastructure for e-puck (and TAM). Technical Report TR/IRIDIA/2015-004, IRIDIA, Université libre de Bruxelles, Belgium, 2015.