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

M. Kegeleirs, D. Garzón Ramos, and M. Birattari
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 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 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 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—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—a logging and replay tool, and rviz—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—

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 System6, 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) GMapping7 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.

6https://github.com/gumstix/yocto-manifest
7http://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](http://wiki.ros.org/multirobot_map_merge)

\(^9\) [http://wiki.ros.org/multimaster_fkie](http://wiki.ros.org/multimaster_fkie)

\(^10\) [http://www.e-puck.org](http://www.e-puck.org)

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


