AMiRo: A Modular & Customizable Open-Source Mini Robot Platform

Stefan Herbrechtsmeier, Timo Korthals, Thomas Schöpping, and Ulrich Rückert

***Abstract*— The AMiRo is a novel modular robot platform that can be easily extended and customized in hardware and software. Built up of electronic modules that fully exploit recent technology and open-source software for sensor processing, actuator control, and cognitive processing, the robot facilitates rich autonomous behaviors. Further contribution lies in the completely open-source software habitat: from low-level microcontroller implementations, over high-level applications running on an embedded processor, up to hardware accelerated algorithms using programmable logic. This paper describes in detail the motivation, system architecture, and software design of the AMiRo, which surpasses state-of-the-art competitors.**

I. INTRODUCTION

Mini robots make their way all into education, research, and commercial use cases to create and teach autonomous systems [1]. With today's microelectronics technology, small sized robots can be built with all functional properties of full sized robots, but with the benefits of affordability and need of little space. These platforms offer a solution for doing robotics in practice at a large scale and with low setup effort besides virtual simulation [2].

Today, multiple solutions which can be seen as mini robots exist. The first concept of a small robot that teachers and researchers could buy at an affordable price was the Khepera I [3]. It was developed and designed by Mondada and colleagues at the École Polytechnique Fédérale de Lausanne (EPFL, Switzerland). This robot has nowadays fallen in disuse and was suppressed over time by its successors up to the latest revision, the Khepera IV [4]. Other developments at the EPFL are the larger s-bot and modular marXbot, which were designed for research purposes only. Commercial products are the e-puck, and the tiny Alice micro robot [5].

There exists a huge gap of small robotic platforms, which offer all functional capabilities of a full sized robot. One can buy a large number of robots with a simple design in processing and mechanics to build up a swarm robotic setup (kilobot [6], I-SWARM [7]), or a single fully autonomous system with high processing power and cutting edge capabilities (PR2 [8]). Additionally, robotic platforms should rather offer an easily extensible and open-source base platform, that enables them to fulfill upcoming tasks, which were not covered by the original design.

The development of the *Autonomous Mini Robot* (AMiRo) had the goal to meet these requirements by creating a fully self-autonomous robot with high endurance and no drawbacks in functionalities due to its small size, combined with a completely open-source software habitat [9]. The design of the AMiRo as described in this paper, draws from the experience gained from former works, as the AMiRo prototype [10], the BeBot mini robot [11], and numerous related student projects. The architectural idea of spreading the computational load over different modules was derived from a modular distributed computer architecture developed at Queensland University of Technology based on Transputers [12]. All software, including the operating systems, applications, and external tools, is open-source and can easily be extended. We hereby present a robotic system, that covers a wide range of applications or offers at least the base platform for swarm, multi, and single robotics after the taxonomy of Iocchi and colleges [13].

Fig. 1: Basic setup (left) and uncovered with extensions (right)

This paper is organized as follows: Section II gives an overview of the system's modular architecture, which comprises the mechanical build-up and electronic interfaces. Section III describes in detail every module of the basic AMiRo setup, before two already developed extension modules are introduced in Section IV. To give a broad overview of its capabilities, already implemented applications are presented in Section V. The last Section VI summarizes the robot's performance data and compares it to other current developments.

II. SYSTEM ARCHITECTURE

The most important feature of the AMiRo is its modular and extensible design. In order to create such a system, a mechanical structure and electrical interface have been defined. This approach allows to enhance the AMiRo's capa-

Bielefeld University, Cluster of Excellence Cognitive Interaction Technologies, Cognitronics & Sensor Systems, Inspiration 1, 33619 Bielefeld, Germany, http://www.ks.cit-ec.uni-bielefeld.de/, {sherbrec, tkorthals, tschoepp, rueckert}@cit-ec.uni-bielefeld.de

bilities with custom extensions that just need to implement the reference design, described in this section. Thus, multi robot applications using heterogeneous systems can be built up with minimal effort.

A. Mechanical Structure

The AMiRo has a cylindric shape with a diameter of $100\,\mathrm{mm}$ and a minimal height of $76\,\mathrm{mm}$, which makes it a very compact robot platform. Its body is fabricated from acrylic glass tube material, roofed with an opaque lid (cf. Fig. 1). Either part is designed to be modifiable, stackable, and exchangeable with custom elements. The chassis covers all electronics, except for one programming port, charging pins, and the power connector (cf. Section III-A and Section III-B). Two wheels with separate gear motors offer a differential kinematic and allow swift movements. At the front and rear centers, metal sliders are attached underneath, stabilizing the robot on flat surfaces.

Inside the chassis, two batteries, the motors and wheels, several sensors, and multiple AMiRo modules (AMs) are located. In addition to the three basic modules (BAMs) an arbitrary number of extensions (EAMs) can be added to the system. Each of these EAMs must have a diameter of $92\,\mathrm{mm}$ or less and may accommodate any hardware from microcontrollers (MCUs) over embedded processors to programmable devices. They are stacked vertically and interconnect through two 60-pin inter AM connectors (IAMCs) at the bottom and top of each module. This way, any combination of EAMs can be stacked in any order and communicate via the common interface (cf. Section II-B). In order to enhance stability, the modules are additionally fixed with four standoff screws each. Fig. 2 depicts how tightly packed the inner parts of the robot are.

B. Electrical Interface

All AMs in the system share a common power and communication interface via the IAMCs with minor exceptions for the basic modules (cf. Fig. 2). It consists of a variable $6\,\mathrm{V}$–$12\,\mathrm{V}$ system power supply, four regulated ones ($1.8\,\mathrm{V}$, $3.3\,\mathrm{V}$, $4.2\,\mathrm{V}$, and $5.0\,\mathrm{V}$), serial communication standards (CAN, UART, and SPI), a parallel camera interface (CI), an external bus interface (EBI), as well as eight dedicated control signals (CTRL). The latter are used to perform the low-level actions “system boot”, “shut down”, and “restart”, activation of additional voltage regulators, and configuration of programmable devices.

All of these interfaces are optional and may just be wired through the module without being used by the accommodated hardware, except for CAN. Its multi-master design and real-time capabilities make it an ideal communication bus for distributed systems like the AMiRo, and thus it is defined mandatory for all AMs.

III. BASIC SYSTEM

In its basic setup, the AMiRo comprises three BAMs and the *ProximitySensor*, a flexible PCB that accommodates several sensors and is integrated in the cylindric part of the chassis. The *DiWheelDrive* module acts as bottom of the robot shell and thus has a diameter of full $100\,\mathrm{mm}$. Instead of IAMCs, two U-profiles are mounted on top of the board on which the *PowerManagement* is attached. The resulting space is utilized to house the batteries, motors, and wheels. These components are enclosed horizontally by the *ProximitySensor*, which is powered and controlled from the *PowerManagement* via a flexible flat cable. Communication between the two lowermost AMs is enabled by another flexible flat cable bridging the gap. Atop the *PowerManagement* an arbitrary number of EAMs can be stacked. Finally, the *LightRing* completes that stack as the last AM just below the opaque lid (cf. Fig. 2).

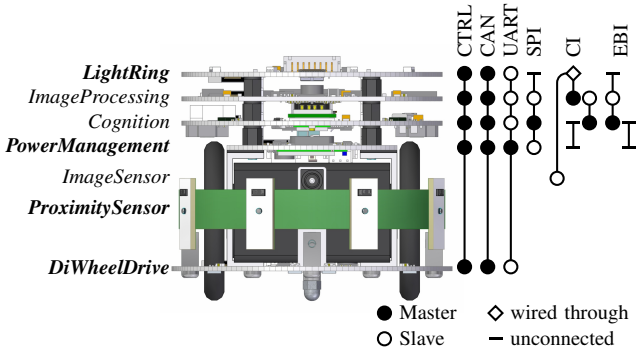


Fig. 2: The inner parts of the AMiRo and an overview of the electrical interfaces with all extensions installed. Modules that are included in the basic setup are printed in bold type.

As the terminology suggests, each BAM is designed to handle specific tasks:

- *LightRing*: status visualization, low-power wireless sensor network, camera interface, and control of an external device
- *PowerManagement*: power supply, power monitoring, battery charging, Bluetooth communication, proximity detection, and user input
- *DiWheelDrive*: motion control, dead reckoning, floor detection, and support for alternative charging

In the remainder of this section, the hardware of these modules is described in detail, as well as the developed software.

A. DiWheelDrive

Equipped with two $1\,\mathrm{W}$ DC gear motors that can be controlled independently, the lowermost AM features a differential kinematic that allows flexible movements. Additionally, multiple sensors such as optical encoders within the motors, as well as a three axis gyroscope, accelerometer, and magnetometer can be used to increase precision of motion control and path integration, and to wake the system from standby mode. Four down facing infrared based proximity sensors with ambient light measurement (VCNL4020) enable the robot to follow lines and to detect cliffs or changes of the ground surface. Since the motors are two of the most power

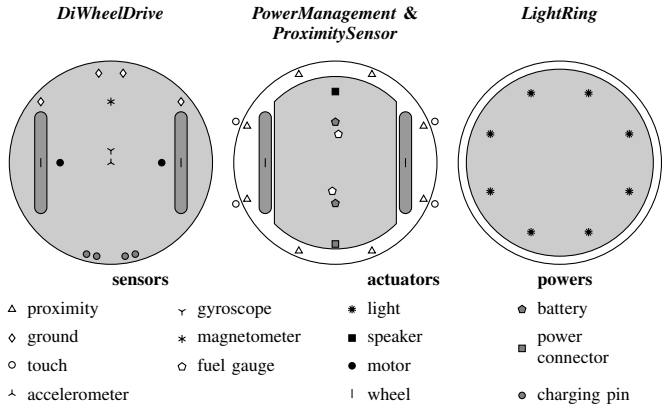


Fig. 3: Top view of the three basic modules showing all sensors, actuators, and power related hardware elements

demanding elements in the system, their current is monitored by two measuring shunts. Two pairs spring mounted pins at the module's rear provide the capability to charge the robot using a station (cf. Fig. 3 and Fig. 4(b)).

In order to perform the required computation, the *DiWheelDrive* is equipped with an ARM Cortex-M3 based STM32F103 MCU from STMicroelectronics. A frequency of 72 MHz combined with 64 kB RAM and 512 kB flash memory allows to calculate motion control, odometry, and dead reckoning algorithms at high sampling rates. It can be accessed either from another AM via the provided communication buses, or using the additional serial (UART) programming port. Although the other basic modules provide such a port as well, only the one of the *DiWheelDrive* is not covered by the chassis. Finally, a 128 byte EEPROM is provided as non-volatile memory.

B. PowerManagement

Being the central module of the basic AMiRo setup, the *PowerManagement* is responsible for power supply and elementary behavior of the system and thus is equipped with the most powerful microcontroller. In contrast to the MCUs on the other BAMs, the ARM Cortex-M4 based STM32F405 from STMicroelectronics features 1024 kB flash memory, a floating point unit, a total of 192 kB RAM, and runs at 168 MHz. This computing power ensures fast reaction times on critical system events and still provides enough headroom to additionally perform basic behaviors like obstacle avoidance, or homing. The MCU is accompanied by a buzzer and a Bluetooth 2.1 + EDR chip. As with the other BAMs, a 128 byte EEPROM is available for storing data permanently.

The *PowerManagement* is extended by several sensors via the *ProximitySensor* module (cf. Fig. 2, Fig. 3, Fig. 4(d), and Fig. 4(e)). It accommodates eight integrated sensors for ambient light and proximity measurement, which are identical to the ones on the *DiWheelDrive*, and four capacitive touch sensors. Whilst the former are equally distributed over the robot's circumference, the latter are located below the two most left and two most right proximity sensors (cf. Fig. 3). All required connections (power, I^2C bus, and interrupt signals) are combined in a single flexible flat cable that links the *ProximitySensor* with the *PowerManagement*.

The eponymous features of this AM are power related capabilities, including charging the batteries, supply the system with current at five different voltages (6 V–12 V, 1.8 V, 3.3 V, 4.2 V, and 5.0 V), and monitoring the corresponding power draws. Therefore, each of the two lithium-ion battery packs is accompanied by a fuel gauge and a charger. The former monitor the status of the batteries and provide information like temperature, state of charge, and remaining uptime. Since the supply voltage can range from 6 V to 12 V, the four regulated ones are generated and monitored by the *PowerManagement* and are available to the system through the IAMCs. For charging the robot, there are two ways to plug the AMiRo to an external power source. First, the *PowerManagment* provides a connector at the rear to plug in a power cord. Second, the robot can dock at a station and charge via the pins of the *DiWheelDrive*.

When the robot is in standby mode, only few components are kept active, so that energy consumption is reduced to a minimum while the system is still responsive to be woken up by external events. Such can occur when a power cord is plugged in, a touch sensor detects a contact or release, or the accelerometer's measures exceed a previously configured value range (for instance when the robot is turned upside down). Each event source can be enabled or disabled before the system enters low-power mode. Thus, the *PowerManagement* can completely turn off the whole AMiRo, except for the wake-up sources and its own MCU, which still operate in power saving mode.

C. LightRing

The *LightRing* is the final AM on top of the module stack. It features eight up facing colored LEDs, that are arranged in a circular manner corresponding to the sensors of the *ProximitySensor* (cf. Fig. 3 and Fig. 4(a)). Each color channel is adjusted individually via a total of 24 dedicated PWMs with 12-bit resolution each, which again are controlled by an STMicroelectronics ARM Cortex-M3 based STM32F103 MCU. There are no IAMCs facing upwards for the lid to be mounted just above the module. As a result, the LEDs are completely covered by the opaque acrylic glass, so that the punctual emitted light is scattered and illuminates the lid (cf. Fig. 1).

Besides its purpose as display, the *LightRing* features a low-power radio module operating at 2.4 GHz and two interfaces to connect additional devices to the AMiRo. Firstly, a parallel CI connector provides direct access to the according bus of the IAMCs. Secondly, an interface combining UART and 5.0 V power supply is provided to attach additional devices to the MCU. Again, an EEPROM can hold 128 bytes of data permanently.

D. Operating System

In order to minimize maintenance effort and risk of errors, the software running on the MCUs is quite similar, regardless

the differing hardware and purposes. It features a hierarchy of four functional layers:

1) bootloader (OpenBLT)
2) real-time operating system kernel (ChibiOS)
3) board specific abstraction layer (AMiRo-OS)
4) high-level behavioral logic (application level)

Whilst the first three layers handle basic functionalities such as hardware access and thread scheduling, the more complex behavioral logic belongs to the application level.

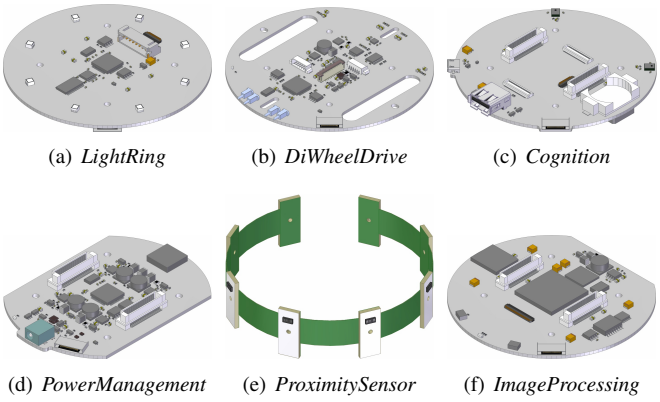
In order to balance flexibility and performance, the four layers are compiled in only two binaries for each AM. Whilst one is dedicated to the bootloader, the other combines all three remaining layers. This way, performance critical code is highly optimized, while the system can be reconfigured easily via the bootloader, as described in the following.

The **OpenBLT** open-source bootloader for MCUs bootstraps the robot's configuration and initializes the operating system [14]. Since the *DiWheelDrive* is the only module of which the programming port can be accessed by the user, new software is applied to the other BAMs by remote flashing via OpenBLT. Using the mandatory CAN bus, the new binary is forwarded by the *DiWheelDrive* so that the target module receives the data and writes it to its MCU's flash memory. The whole toolchain for flashing the BAMs is encapsulated in this project.

ChibiOS is another open-source project that aims at creating a portable, light weight, and fast real-time operating system for embedded applications [15]. In addition to low-level drivers for all interfaces of the most common microcontrollers, it provides a unified hardware abstraction layer, priority based preemptive scheduling, as well as high-performance event and message passing systems. Its several modules can be accessed or expanded by using the provided C and C++ interfaces, while the whole kernel is completely transparent and offers access to all functionalities. As a result, ChibiOS can be configured statically and dynamically to exactly fit the application in order to maximize efficiency and performance.

AMiRo-OS is the operating system running on all MCUs of the AMiRo and is completely open-source as well [16]. It configures the underlying ChibiOS kernel and extends it with specific hardware drivers, depending on the AM. For further extension of the software by high-level applications, it provides abstract C++ interfaces for all hardware drivers, most of which are running in individual threads. Since context switching is computationally cheap with ChibiOS, such a multi-threaded approach has the advantage of utilizing the integrated scheduler for priority based hardware access.

In order to synchronize the system status on all AMs, AMiRo-OS provides an unified data structure, which is identical for all instances. Each module periodically broadcasts its current status via CAN, so that all other AMs can update their local data. Thereby, the system is inherently fail-safe, since hardware or software related issues result in absence of expected transmissions, which can be detected by each AM. In addition to the periodic communication, request commands can be used to trigger specific actions or to realize remote procedure calls.



(a) *LightRing* (b) *DiWheelDrive* (c) *Cognition*

(d) *PowerManagement* (e) *ProximitySensor* (f) *ImageProcessing*

Fig. 4: Current set of AMiRo modules

IV. EXTENSION MODULES

In addition to the three basic AMs, three extensions have been developed so far. First, the *ImageSensor* module hosts a 5 megapixel camera and provides its electrical interface via CI (cf. Fig. 2). Second, the *Cognition* and *ImageProcessing* AMs extend the AMiRo platform by an embedded processor and programmable logic.

A. Cognition

The *Cognition*'s most important feature is the attached Gumstix Overo Computer-on-Module (COM). In the current setup we use the Overo TidalSTORM COM, which combines an ARM Cortex-A8 based DaVinci DM3730 SoC from Texas Instruments running at 1 GHz with 1 GB RAM and a microSD card as hard drive. Actually, any Overo COM can be mounted on the base board via two 70-pin connectors for future upgrades. Additionally, the camera interface of the IAMCs can be accessed by using a flexible flat cable that links the *Cognition* to the Overo COM.

This EAM hosts several further wired and wireless communication interfaces, the Overo COM can utilize. First, there is the same serial programming port as on the basic AMs. Second, three USB interfaces (one Type A, one micro-AB, and an internal pin header) can be used to attach external devices. Third, a wireless module provides 802.11b/g/n and Bluetooth 2.1 + EDR capabilities.

Two speakers combined with two microphones enable bidirectional sonic communication and allow localization of an acoustic source. As with the basic modules, a 128 byte EEPROM is available. Finally, a battery holder for a coin cell can be used to provide power to the system clock of the SoC even when the AM is powered off via the IAMCs (cf. Fig. 4(c)).

The operating system running on the Overo COM is based on the Linux Yocto reference distribution Poky [17]. Build recipes are provided open-source, so that all possible software approaches and ideas can be realized later on

[18]. For interfacing other AMs via CAN, a simple to use SocketCAN software-wrapper is offered.

B. ImageProcessing

The *ImageProcessing* was developed with the goal in mind to increase efficiency and performance of image- and video processing. It hosts a Xilinx Spartan 6 FPGA, accompanied by a $128\,\mathrm{MB}$ LPDDR-SDRAM. In order to allow permanently storing of data, an $8\,\mathrm{MB}$ flash memory is provided as well. The FPGA has access to all communication buses of the IAMCs for most flexible use. In addition to the *ImageSensor*, up to three more cameras can be connected directly to this EAM via a 51 pin ZIF connector, using either parallel or serial camera interfaces (cf. Fig. 4(f)). Although this EAM was primarily designed for image processing purposes, it is up to the user to run any other computation on the FPGA.

V. APPLICATIONS AND TOOLS

The **AMiRo-OS** includes two basic applications, namely a rudimentary command shell and a hardware test. The former is already integrated in ChibiOS and is extended by module specific commands. The latter is such a callable function and performs a test on all periphery elements of the AM. Additional commands provide calibration functions for sensors, access to the EEPROM, and basic information about the system.

The ***DiWheelDrive*** offers the fundamental interfaces for moving in the environment and sensing the motion. For steering with the differential kinematic, the "driving straight" motor controller by Thomas Bräunl [19] is implemented. To handle constructional disparities in the differential drive, an interface for setting calibration parameters for the base width and wheel diameters, as introduced by Borenstein and Feng [20], is available. The Euler-Collatz odometry is implemented and extended by a Kalman filter, including measurements from the accelerometer and gyroscope. Via steering commands either velocities or target positions relative to the robot can be set. Furthermore, the *DiWheelDrive* can be triggered to autonomously dock to a charging station.

Since the ***PowerManagement*** features the most powerful MCU of the BAMs, it handles computationally intensive tasks, such as detecting the environment and acting on it with a local planner strategy. The environment detection is done by an extended inverse particle filter model as introduced by Kleppe and Skavhaug [21], taking all proximity sensors into account to get a detailed surround view. Based on this approach, a local planner is able to receive target positions which are pursued by sending the appropriate commands to the *DiWheelDrive*.

The ***LightRing*** offers, besides the status visualization via its eight colored LEDs, a serial breakout which can be used to interface additional sensors like the Hokuyo URG LX04 laser scanner. An implementation of the tinySLAM algorithm by Steux and Hamzaoui [22] is realized for simultaneous localization and mapping.

The ***Cognition*** sums up all high-level tasks to gather and coordinate information offered by the other AMs. For all common tasks the open-source middleware RSB [23] is used, but the Robot Operating System (ROS) is supported as well. In addition to the abstraction of all available sensors and actuators across the robot, the open-source *MuRoX* project includes numerous tools and applications which are exclusively running on the *Cognition*'s COM [24]. Besides the basic applications, which are a full SLAM approach, a WebSockets based web server for teleoperation, and multi robot exploration by Yamauchi [25], we are concentrating in applying tabletop tasks concerning the interaction with humans, namely the *General Purpose Tabletop Robotics*. This field first emerged with the participation in the *RoboCup@Home* league 2014 in Magdeburg, Germany together with ToBI [26]. The applications comprise the human interaction of small, handy robots in a work or household environment. As part of this, robots are acting on their own or cooperate to detect and accomplish tasks like cleaning up, delivering, or free human interaction.

The ***ImageProcessing*** provides high parallelization capabilities and thus is used for acceleration of image- and video processing. It applies filters like edge- and corner detectors in real time on the latest data from the camera stream. Via EBI, the results can be read by the Overo COM on the *Cognition* directly from the FPGA. This way, the *ImageProcessing* acts as a preprocessor that releases the SoC from high computation load. Furthermore, this EAM is used as a demonstration platform for new processing architectures, such as the resource efficient VLIW processor CoreVA [27].

The **Gazebo simulator** is used for exhaustive or experimental applications, thus the AMiRo and its kinematic properties have been ported to several models. For now there exist the base setup, a version with installed Hokuyo URG 04LX, and a Kinect mounted one. Abstraction between simulation and real hardware exists at the RSB interface level. The interfaces of all integrated sensors and actuators have been ported from the real robot to Gazebo, so that user applications which run in the simulator can seamlessly be transferred and executed on the *Cognition*'s COM, or with minor modifications on any MCU based AM.

VI. PERFORMANCE CHARACTERISTICS

This chapter summarizes the AMiRo platform and compares it to its current competitors. All robots in Table I are chosen with respect to their computational capability, size, price, and age of no more than six years. They all incorporate a CPU which can at least run a full UNIX-like operating system to execute complex software like ROS. This feature makes them superior over swarm robots, which commonly host reduced bio-inspired logic, and compliant to virtually any robot used in science and education. Additionally, WLAN, USB, RGB camera, speaker, and microphones are supported by all listed systems.

The AMiRo hosts the most powerful processing units comprising the MCUs and CPU, as well as the programmable logic, which is unique among the compared devices. With its $29\,\mathrm{Wh}$ accumulators, the AMiRo can last $13\,\mathrm{h}$ with all EAMs attached doing no application calculations and movements,

TABLE I: Performance characteristics of two-wheeled mini robots

	∅ × H (mm)	Energy (Wh)	CPU @ Freq. (MHz)	RAM (MB)	Programmable Logic	3-Axis Accelerometer	3-Axis Gyroscope	#Proximity Env.	#Proximity Gnd.	Open-Source Software
e-puck	75x60	5.92	TI AM3703 @ 800	512	–	Yes	No	10	3	Yes
marXbot	170x290	38.16	Freescale iMX31 @ 533	128	–	Yes	Yes	24	12	No
K-junior	125x150	4.44	TI OMAP3503 @ 600	512	–	Yes	No	6	4	No
Khepera IV	140x58	25.16	TI DM3730 @ 800	512	–	Yes	Yes	8	4	No
AMiRo	100x94	28.86	TI DM3730 @ 1000	1024	XC6SLX150	Yes	Yes	8	4	Yes

or $6\,\mathrm{h}$ with full utilization of MCUs, CPU, camera, and locomotion (oscillating acceleration with two second period and $600\,\mathrm{mm/s}$ target velocity). When using only the basic setup, the uptime increases to $41\,\mathrm{h}$ and $12\,\mathrm{h}$ respectively.

The robot already approved as a teaching platform in the lectures "Autonomous Systems Engineering", "Processor Architectures" with robotic use cases, and student projects in "Intelligent Systems". The latter include participation in the *RoboCup@Home* league in collaboration with ToBI, which resulted in a well situated constitution. All software and build environments are open-source, as well as current and upcoming demonstrations. Therefore, a huge set of runnable applications is available for easy understanding of the whole AMiRo habitat.

Development of the AMiRo has not ended, but will continue to further enhance the system. An upcoming modification substitutes the *DiWheelDrive* by a new locomotive device to give the AMiRo the ability to fly. In order to provide capabilities of manipulating the environment, the basic platform is planned to be extended with additional actuators. Furthermore, the current modules will be revised as well, to keep the hardware up to date.

ACKNOWLEDGMENT

This research/work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

REFERENCES

[1] U. Rückert, S. Joaquin, and W. Felix, *Advances in Autonomous Mini Robots: Proceedings of the 6-th AMiRE Symposium*. Springer Publishing Company, Incorporated, 2012.

[2] U. Rückert, J. Sitte, and U. Witkowski, Eds., *Autonomous Minirobots for Research and Edutainment*. Heinz Nixdorf Institut, Universität Paderborn, 2007, vol. 216.

[3] F. Mondada, E. Franzi, and A. Guignard, "The development of khepera," in *Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop*, 1999, pp. 7–14.

[4] K-Team. (2016-02-27) Khepera IV. [Online]. Available: http://www.k-team.com/mobile-robotics-products/khepera-iv/

[5] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robotics*, vol. 2013, pp. 1–10, 2013.

[6] M. Rubenstein, C. A., and N. R., "Programmable self-assembly in a thousand-robot swarm," Berlin, Germany, 2014.

[7] J. Seyfried, M. Szymanski, N. Bender, R. Estaña, M. Thiel, and H. Wörn, "The i-swarm project: Intelligent small world autonomous robots for micro-manipulation," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. Şahin and W. Spears, Eds. Springer Berlin Heidelberg, 2005, vol. 3342, pp. 70–83.

[8] E. Berger and K. Wyrobek. (2016-02-27) PR2. [Online]. Available: http://www.willowgarage.com/pages/pr2/overview

[9] T. Schöpping, T. Korthals, S. Herbrechtsmeier, and U. Rückert, "AMiRo: A Mini Robot for Scientific Applications," in *Advances in Computational Intelligence*, I. Rojas, G. Joya, and A. Catala, Eds., vol. 9094. Springer International Publishing, 2015.

[10] S. Herbrechtsmeier, U. Rückert, and J. Sitte, "AMiRo – Autonomous Mini Robot for Research and Education," ser. Advances in Autonomous Mini Robots, U. Rückert, J. Sitte, and F. Werner, Eds. Springer, 2012, pp. 101–112.

[11] S. Herbrechtsmeier, U. Witkowski, and U. Rückert, "BeBot: A Modular Mobile Miniature Robot Platform Supporting Hardware Reconfiguration and Multi-standard Communication," vol. 44. Springer, 2009, pp. 346–356.

[12] M. K. and S. J., "Continuous action space reinforcement learning on a real autonomous robot," ser. HNI-Verlagsschriftenreihe, Heinz Nixdorf Institut, 2001, pp. 151–159.

[13] L. Iocchi, D. Nardi, and M. Salerno, "Reactivity and deliberation: A survey on multi-robot systems," in *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, vol. 2103, pp. 9–32.

[14] F. Voorburg. (2016-02-27) OpenBLT. [Online]. Available: http://feaser.com/en/openblt.php

[15] G. Di Sirio. (2016-02-27) ChibiOS. [Online]. Available: http://chibios.org

[16] S. Herbrehtsmeier, T. Schöpping, and T. Korthals. (2016-02-27) AMiRo-OS. [Online]. Available: http://opensource.cit-ec.de/projects/amiro-os/

[17] T. Graydon and B. Flanagan. (2016-02-27) Poky Linux. [Online]. Available: http://www.pokylinux.org/

[18] S. Herbrechtsmeier and T. Korthals. (2016-02-27) meta-openrobotix. [Online]. Available: http://opensource.cit-ec.de/projects/meta-openrobotix/

[19] T. Bräunl, *Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems*, 2nd ed. Springer, 2006, pp. 65–66.

[20] J. Borenstein and L. Feng, "Correction of systematic odometry errors in mobile robots," *Proceedings of the 1995 International Conference of Intelligent Robots and Systems (IROS'95)*, pp. 569–574, Aug. 1995.

[21] A. L. Kleppe and A. Skavhaug, "Obstacle detection and mapping in low-cost, low-power multi-robot systems using an Inverted Particle Filter," M. Roy, Ed., Toulouse, France, Sep. 2013, p. NA.

[22] B. Steux and O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program," in *ICARCV*. IEEE, 2010, pp. 1975–1979.

[23] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," in *System Integration (SII), 2011 IEEE/SICE International Symposium on*, Dec 2011, pp. 1183–1190.

[24] T. Korthals and T. Schöpping. (2016-02-27) MuRoX. [Online]. Available: http://opensource.cit-ec.de/projects/murox/

[25] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the Second International Conference on Autonomous Agents*, ser. AGENTS '98, 1998, pp. 47–53.

[26] S. Meyer zu Borgsen, T. Korthals, L. Ziegler, and S. Wachsmuth, "ToBI-Team of Bielefeld The Human-Robot Interaction System for RoboCup@ Home 2015," 2015.

[27] B. Hübener, G. Sievers, T. Jungeblut, M. Porrmann, and U. Rückert, "CoreVA: A Configurable Resource-efficient VLIW Processor Architecture," in *Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing*. IEEE, 2014.