Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-27 (2019) 192–197

Design of a Prototype for a Modular Mobile Robotic Platform

Milan Tkáčik, Adam Březina, Slávka Jadlovská *

* *Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, Košice 042 00, Slovakia (e-mail: milan.tkacik@tuke.sk, adam.brezina@tuke.sk, slavka.jadlovska@tuke.sk)*

Abstract: This paper deals with the design and implementation of a prototype for a modular mobile robotic platform for the universal use in both research and teaching activities. It describes the development of a mobile robot skeleton which is low-cost and easy to produce by means of 3D printing. It next deals with prototyping the robot's hardware components, focusing on the development of the robot's main control unit and implementation of communication interfaces. The concept of multiple sensor modules connectable to the robot's top side is also described. The paper next concentrates on the implementation of program modules at the level of MicroController Unit (MCU) on the control board, including the management of peripheral components as well as the implementation of control algorithms and communication protocols. Last but not least, it deals with the development of software modules for the MCUs on individual sensor modules and with the implementation of services for robot control at the level of the supervisor computer using the Robotic Operation System (ROS) communication system.

Keywords: Autonomous mobile robots, Chassis control, Functional blocks, Image recognition, Measuring units, Mobile robots, Motor control, Velocity control, Universal platform

1. INTRODUCTION

Nowadays, there is a large number of mobile robotic applications and application-specific mobile robots exist for almost any of them, like line follower robots, robotic soccer players or maze solver robots. However, it is difficult to find a mobile robotic platform suitable for use in multiple application types without the need for major hardware and software adjustments (Yim et al. (2002)). For this reason, several groups have begun to develop modular mobile robots or robots with the possibility of self-configuration and self-replication. An example of a self-configurable robot is the M-Blocks project at MIT (Romanishin et al. (2013)), which can change its shape using pulses of angular momentum and magnetic hinges. Another approach of modular robotics is VSA-CubeBot, which offers a customizable platform for the realization and test of variable stiffness robotic structures with many degrees of freedom (Catalano et al. (2011)).

The Center of Modern Control Techniques and Industrial Informatics (CMCT&II) at DCAI, FEEI, TUKE has been dedicated to the development and research of mobile robotic systems since its establishment (Jadlovský et al. (2016)). The mobile robotic platforms developed within CMCT&II include a robotic soccer player constructed according to the MiroSot category specifications (Jadlovský and Kopčík (2016)). It is a two-wheel differential-drive mobile robot whose primary use is for robotic soccer, but after slight modification it can also be used for other applications, such as finding the path in a maze (Jadlovský et al. (2016)). Another mobile robot developed in CMCT&II is a tracked mobile robot suitable for line-following or obstacle avoidance applications (Jadlovská et al. (2016)).

This paper presents the design of a prototype for a novel modular robotic platform as an embedded system (Jadlovská et al. (2016)). It is a small tracked autonomous robot with the possibility of connecting any sensors/actuators to the robot itself. This approach ensures that the same robot body can be used for various applications without the need to modify the hardware or software concerning the robot's control board. The robot can be used for a number of applications in cybernetics including navigation, image recognition or intelligent control algorithm verification (Santos et al. (2013)).

The paper is divided into two main sections. First of these deals with the design and construction of the robot's skeleton and the main control unit, and with the introduction of several sensor modules to verify the concept of robot's modularity. The second section addresses the software development part, focusing on the implementation of program modules for the main control unit of the robot, as well as for individual sensor modules. Implementation of services for robot control at the supervisor computer level through the Robotic Operation System (ROS) system is also described.

2. HARDWARE DESIGN

The robot has a unique design to ensure the modularity of individual components. The skeleton was designed via Autodesk Fusion 360 CAD software (Barrie et al. (2016)),

10.1016/j.ifacol.2019.12.755

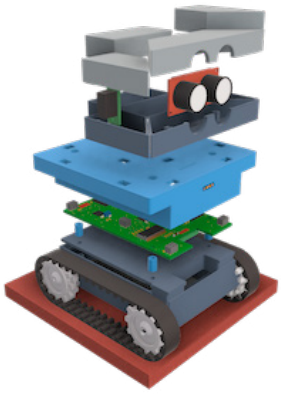


Fig. 1. 3D render showing disassembled skeleton of robot

and Autodesk Eagle CAD software was used to create the printed circuit board (PCB) designs (Bailey et al. (2017)). The final design is based on merged results of these two software systems using the Autodesk Cloud and PCB 3D model export directly to the Fusion 360 project.

2.1 Skeleton of the robot

The skeleton of the robot is made up of several plastic parts printed on a 3D printer. Printing time for the robot took around 16 hours, with the layer height of 0.1 mm (Cellucci et al. (2017)). The robot's drive is provided by two Pololu micro-motors with 1:210 ratio gearbox together with Pololu tracks. To determine the motor speed, two Pololu magnetic encoders with resolution of 12 pulses per revolution are mounted directly on the motor shafts. The lower part of the chassis serves as a storage space for a two-cell Li-Poly battery that supplies power to all parts of the robot, modules included. The cover of the lower part of the skeleton also secures the mounting of the robot's control unit as well as the top cover, to which the modules are attached using six-pin connectors. The way the individual parts are interconnected can be seen in Fig. 1.

2.2 Main control unit

The main control unit consists of the robot's control board and the Raspberry Pi as a supervisor computer (Vanitha et al. (2016)). It was designed to provide all necessary functionality to control the movement of the robot and to communicate with the supervisor computer and the individual modules (Tkáčik (2019)). The diagram that depicts the interconnection of individual parts of the main control unit can be seen in Fig. 2.

The control unit's power supply provides stable power for all parts of the electronics, including the MCU, micro-motors driver and Raspberry Pi supervisor computer. Several power regulators were used for this purpose including 5 V switching regulator, 3.3 V and 5 V linear regulators. A resettable fuse with a power diode is located directly at the power connector. Such wiring ensures the protection of the electronics against polarity reversal or short circuit on the control board.

The control of the micro-motors is realized by the L298 H-Bridge driver that switches the individual channels fed to the motors based on the signals from the microprocessor.

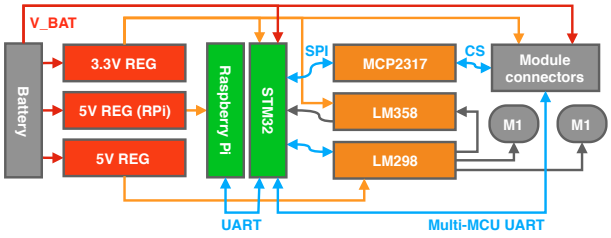


Fig. 2. Connection of individual components of the robot

This driver also provides the capability to measure the current of each motor using analog converters within the main microprocessor.

The central control unit of the entire board is the STM32F103 MicroController Unit (MCU) provided by the STMicroelectronics. It provides control of all peripheral devices on the control board and it also provides communication with the supervisor Raspberry Pi computer and also with individual modules via serial buses.

A UART bus in a multiprocessor communication mode is used to communicate with expansion modules. On the robot's control board, there are eight connectors for the modules, but their number can be increased up to twelve by connecting two extension side boards. Each connector has both UART lines, battery power supply to power a more energy-consuming module, 3.3 V power line to power the microprocessor in the module and a chip-select.

The chip-select can be used in two modes: either the main microprocessor determines which connector should be active or it is determined on which connector is the module with a specific address connected. Chip-select is implemented using the SPI programmable I/O expander MCP2317. The complete design of the main control unit that was next assembled can be seen in Fig. 3.

As the robot's supervisor computer, a Raspberry Pi Zero W single-board computer is used. It is mounted directly on the control board and communicates with the main microprocessor via an asynchronous serial link. The Raspberry Pi computer has a built-in WiFi and Bluetooth interfaces, allowing wireless connection to the robot without the need of any additional hardware (Vanitha et al. (2016)).

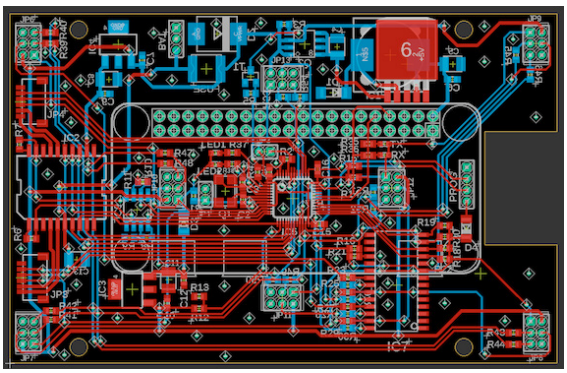


Fig. 3. Design of the main control unit

2.3 Design of the expansion modules with sensors

As mentioned above, the prototype of the presented mobile robot involves the use of multiple different modules, which may include any kind of sensors or actuators. Four types of modules have been developed to verify this concept: the *ultrasonic distance sensor module*, the *infrared proximity sensor module*, the *gyroscope position refinement module* and the *camera module*.

The role of the STM32F051 MCU is to maintain the operation of individual sensors and the communication with the main microprocessor on the control board. For each sensor type, a printed circuit board containing the above-mentioned microprocessor was designed, with each type of module utilizing some of the microprocessor peripherals.

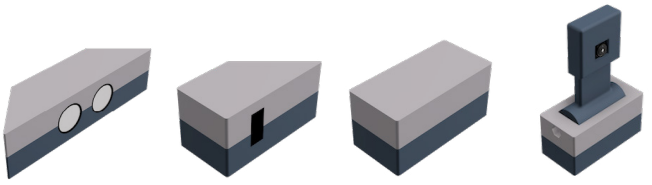
The *ultrasonic distance sensor module* can be used to measure the distance between the robot and an obstacle. The SEN-13959 ultrasonic sensor uses an MCU's built-in timer to measure the duration of the received pulse from the sensor. The measurement of the received pulse length is performed by an external interrupt when the built-in timer is triggered between arrival of the rising edge and value is stored upon arrival of the falling edge. Based on the obtained value, the distance from the sensor to the obstacle is calculated. The 3D model of this module can be seen in Fig. 4a.

The *infrared proximity sensor module* uses the Pololu-1132 infrared proximity sensor, which is connected to the digital input of the module's MCU. The sensor is able to detect an obstacle in the range of 5 to 15 cm. If the obstacle is within this range, the output of the sensor is the voltage value of the logic one, otherwise the logic zero. The 3D model of this module can be seen in Fig. 4b.

The *gyroscope position refinement module* uses a triaxial gyroscope, accelerometer and magnetometer OKY3231. It is connected to the module's MCU by I2C bus. and uses built-in analog converters that operate with a fixed sampling period to determine the linear and angular accelerations in each axis. The measured values are sent to the module's MCU periodically. The 3D model of this module can be seen in Fig 4c.

The *camera module* includes the Raspberry Pi Camera Module v2 with 8-megapixel sensor. Unlike other modules, the camera module is connected directly to the Raspberry Pi Zero W computer using the CSI bus. This module can be used for many applications concerning computer vision. The 3D model of this module can be seen in Fig. 4d.

After completing the design of all modules, the printed circuit boards were manufactured and assembled. MCUs on individual boards were programmed with a different address assigned to each. After testing the individual PCBs, these boards were placed in boxes of individual modules printed on a 3D printer. Printing time was approximately 1.5 hour per one module with layer height 0.1mm. The resulting modules can be seen in Fig. 5.



(a) Sonar (b) Infrared (c) Gyroscope (d) Camera

Fig. 4. 3D models of expansion modules for robot

3. SOFTWARE DEVELOPMENT

Software modules developed for this robot consist of several parts. First of these is the firmware for the main microprocessor on the control board, which is written in C programming language. The second part represents the firmware for the sensor modules, whose main part is the same for all modules and the custom part differs according to the specific sensor type. The third part is the set of program modules written in C++ language using the ROS system in the Raspberry Pi supervisor computer (Quigley et al. (2009)).

3.1 Control board microprocessors firmware

The operation of individual robot peripherals is provided by several program modules written in C language using the Keil uVision 5 IDE. The program is uploaded to the main microprocessor via the control board's service connector using the ST-LINK V2 programming adapter.

The *GPIO program module* provides the initialization of MCU's I/O ports, as well as features to control the Raspberry Pi's power supply. It also monitors the main battery voltage and turns off the power supply when the voltage drops below a critical value and puts the microprocessor in a low power mode.

The *MBPROTOCOL program module* provides a packet that the microprocessor uses to communicate with the supervisor Raspberry Pi computer. This packet consists of five parts, a 16-bit header for identifying the requested command, an 8-bit identification number for addressing attached modules, two 32-bit data fields and an 8-bit control checksum as seen in Fig. 6. The *MBPROTOCOL module* also provides functions for calculation and verification of the packet's checksum.

The *UART program module* provides communication with Raspberry Pi over an asynchronous serial link with a transfer rate of 115,200 bps. This communication uses the above-mentioned packet. Both communication directions are handled by DMA channels, which greatly relieves the processor from processing individual requests within the interrupts. The interrupt is triggered just when the entire

Fig. 5. Completed expansion modules

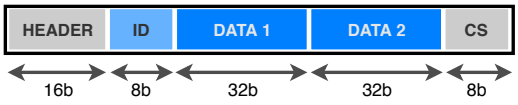


Fig. 6. Packet used to communicate with supervisor PC

packet is received. At this point in the interrupt handler, the packet header determines which module the message is intended for. Subsequently, the received message is passed to the program module and a response packet is generated based on the data received from the program module, which is sent back to the Raspberry Pi using DMA.

The *ENCODERS program module* is used to initialize timer to the quadrature encoder mode. It also provides functions for counting the number of pulses from the timer register followed by clearing the register. Based on the read values, the exact wheel speeds are calculated in the *DRIVE program module* (Santos et al. (2013)).

The *DRIVE program module* is used to control the speed of individual motors by changing the duty on the PWM channels. It also calculates the speed and position of the robot based on odometry, while taking encoder count data from the *ENCODERS program module*. At the same time, it ensures the regulation of the robot's speed by means of the feedback controller implemented in the *CONTROLLER program module*. The *CONTROLLER program module* provides functions for calculating the voltage for motors based on the desired and measured wheel speed while supporting PSD and polynomial controller parameters (Čerkala and Jadlovská (2017)).

The *MOTION program module* implements another P controller in cascade with the wheel speed controller. This controller is used to regulate the position of the robot to the desired coordinates. The controller is not active during the entire program run, but only if the *UART program module* receives a request to move the robot to a specific position. When the desired position is reached, the controller switches off automatically.

The *TIMER program module* provides initialization of timer 1 to generate an interrupt every 20 ms. Functions that require their regular execution to operate are called from this interrupt handler, such as calculation of the next control step in the *DRIVE* and *MOTION program modules* or measurement of the battery voltage.

The *IOEXPANDER program module* provides initialization and operation of the I/O expander on the robot's control board. At the same time, it provides functions for selecting the module connector with which the microprocessor is to communicate (in chip-select mode). The second option to use this program module is to find out on which connector the module with the given address is connected.

The *UART2 program module* ensures the communication with connected modules. Since all modules are connected to one bus, they must be distinguishable by their address. The communication uses the UART protocol in multiprocessor communication mode, where the sent words are 9-bit long, with the set highest bit indicating the address byte or the reset highest bit is the data byte. To initiate the connection, the main microprocessor sends a word to the bus containing the module address with which it requires to communicate and then the entire data exchange is executed with a module that recognizes its address.

The simplified algorithm of message processing within the main MCU can be seen in Fig. 7. The MCU is woken up on whole message reception from the Raspberry Pi using the UART bus, when an interrupt function is invoked. Firstly the checksum of the message is checked. If it doesn't match the calculated checksum, then an error message is sent back to the Raspberry Pi.

Upon successful reception of the message, a command is evaluated based on the header of the packet. The command can be dedicated to either motion control or to the individual modules connected to the robot. If the command is dedicated to motion control, then the required speed or position is set to required motion controller. On the other hand, if the message is for the sensor module, then it is forwarded to the specific module based on its address specified in the ID field of the packet.

In both cases, after the execution of the command, a response is evaluated and inserted into a new response packet. Finally, the response packet is transmitted back to the Raspberry Pi using the UART bus.

3.2 Communication between modules and main MCU

Since any number of hardware modules can be connected to the robot (theoretically up to 254), it was necessary to implement a certain way of addressing the individual modules, since they are all connected to the main microprocessor by one serial bus. After considering a number of options, it was chosen to implement the asynchronous UART communication protocol with the possibility of multiprocessor communication (UART 2 in Fig. 9).

In this mode, the main microprocessor behaves like a master, so it does not have its address and initiates all communication on the bus. Each module is assigned its own 8-bit address when uploading a program, on the basis of which each module knows whether the message transmitted over the bus is intended for it or not. The UART in all microprocessors is set to a baud rate of 115,200 bps using 9-bit word length, no parity, and one stop bit.

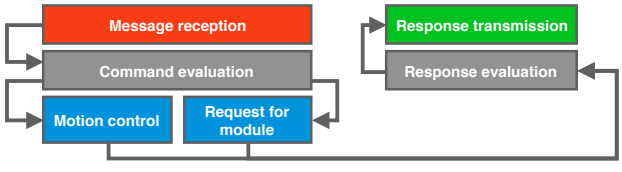


Fig. 7. Command execution algorithm (main MCU)

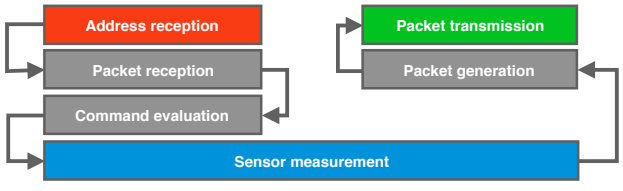


Fig. 8. Command execution algorithm (module's MCU)

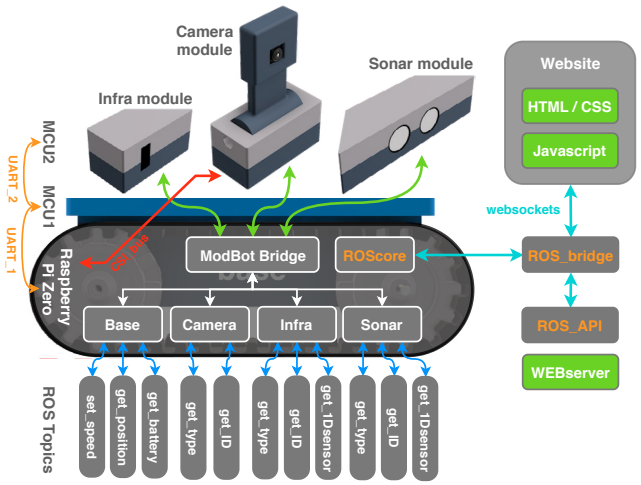


Fig. 9. Connection of individual modules within the robot

At the beginning of the communication, the Master sends the module address to the bus it wants to communicate with. Subsequently, the master sends four bytes containing the request data. A module whose address matches the sent address to the bus begins to receive the data and upon receipt of all four bytes of the command, an interrupt is generated. In this interrupt, it evaluates the command received from the Master, performs the required measurement or setting and in response, sends four data bytes to the bus containing the measured data or a confirmation of command execution. If the addressed module does not respond within a specified time period, it is considered as disconnected. The actual robot with connected sensor modules can be seen in Fig. 10.

The simplified algorithm of message processing within the module's firmware is shown in Fig. 8. Firstly, an address byte is received. If the received address matches the module's own address, then upon reception of all remaining bytes of the message an interrupt is generated.

Within the interrupt handler, the received command is evaluated. Based on the command, the data are either set or read from the sensor. When the read command is received, then a measurement of desired value is initiated and upon its completion, the measured value is converted to actual physical unit. Finally, the converted value is inserted into the response packet which is sent back to the main MCU. If an error occurred during the measurement, then instead of a measured value an error message is sent within the response message.

3.3 Software in the Raspberry Pi supervisor computer

To create a simple abstraction of all hardware parts of the robot and additional modules, the ROS platform was used (Quigley et al. (2009)). Linux Raspbian image with precompiled ROS and OpenCV libraries (Culjak et al. (2012)) was used as the Operation System for Raspberry Pi Zero-W. Using the ROS system, development of remote visualization or remote robot control is straightforward.

To communicate with the main microprocessor with Raspberry Pi, another asynchronous UART bus is used. This bus is connected to the Raspberry Pi hardware serial line. The UART is configured to communicate at 115,200 bps with 8-bit word length, no parity and one stop bit. For the UART data transmission, the above-mentioned packet is used, whose structure can be seen in Fig. 6. Communication is always initiated by the Raspberry Pi, where on each request packet the main microprocessor responds with one answer packet.

The connection between the Raspberry Pi's serial line with the ROS system is implemented in the C++ language within the *ModBot_Bridge program module* (Quigley et al. (2009)). This program module uses the created *Serial class* to transfer individual packets between the Raspberry Pi and the microprocessor on the control board. After the creation of the *Serial object*, the hardware serial line on the Raspberry Pi is initialized with the required parameters. If one of the initialization parts fails, the module provides an error message and program terminates.

The *ModBot_Bridge program module* provides the *service packetSrv* within the ROS system, through which other ROS modules can initiate communication with the robot for the purpose of sending or reading data, see Fig. 9. The ROS *service packetSrv* uses the above-mentioned *Serial class* to communicate with the robot's microprocessor, implementing a blocking mechanism to ensure the transaction's atomicity. Using this mechanism, it is possible to avoid situations where another packet would be sent to the bus before receiving the response to the first packet.

The *Base program module* was created to make abstraction of all robot functionality in the ROS platform (Březina (2019)). Subscribe services for setting wheel speed, setting linear and angular speed of robot and setting waypoints were created. Publish services were also created to enable getting the battery level value, relative position of the robot and wheels speed values. The *MODI_Base module* is connected with the ROS *service packetSrv* to *MODI_Bridge* to obtain all data from control board.

The remaining program modules provide the abstraction of functionalities of each sensor module: *Sonar, Infra, Camera and Gyro program module*. All these program modules use the *packetSrv service* to obtain the data from sensor modules via the control board, as can be seen in Fig. 9. Upon receiving the data, interesting values are published using ROS messages.

When used in different combinations, the provided program modules can be used to implement a variety of robotic applications; some basic ones are listed in Tab. 1. The robotic platform can also be used in the educational process to teach microcontroller programming, OS Linux

Fig. 10. Fully assembled prototype with sensor modules

Table 1. Modules required for basic applications

Application	Modules	Roles
Line follower	Sonar Camera	Obstacles detection Line detection
Ball tracker	Camera	Ball detection
Maze solver	Sonar Infra Gyroscope	Front wall detection Side walls detection Position refinement
Robotic soccer	Bluetooth (RPi) Gyroscope	Communication Position refinement

or circuit design and other activities which involve creating new modules and testing different algorithms, with unique combinations of expansion modules for every application.

4. CONCLUSION

The aim of this paper was to provide a thorough presentation of the design of a modular tracked robotic platform prototype at the CMCT&II at DCAI, FEEI, TUKE. This tracked robot is an embedded system which can be used for multiple applications from robotic soccer to maze solving without the need to modify the MCU firmware, just by building program modules using the ROS system. The design of the robot allows for the simple development of additional modules as well as connecting multiple modules at once, which facilitates the customization of the robot for the needs of a specific user application. The motion control system in the robot's MCU provides stable regulation of the desired robot's speed or position. Further modifications of the robot will include a more powerful supervisor computer with the aim to allow the user to create more complicated applications.

The presented robotic platform is planned to be used for CMCT&II research activities which include verification of advanced control algorithms, implementation of swarm robotics applications or non-destructive diagnostics. It can be also used in the teaching process to demonstrate practical applications that involve programming microcontrollers, creating complex applications using a variety of sensors or practicing the implementation of different control algorithms.

ACKNOWLEDGEMENTS

This work has been supported by the grant KEGA - Implementation of research results in the area of modeling and simulation of cyber-physical systems into the teaching process - development of modern university textbooks - 072TUKE-4/2018.

REFERENCES

C. Bailey et al. Augmenting computer-aided design software with multi-functional capabilities to automate multi-process additive manufacturing. *IEEE Access*, 6: 1985–1994, 2017.

J. Barrie et al. Applications for cloud-based cad in design education and collaboration. In *DS 83: Proceedings of the 18th International Conference on Engineering and Product Design Education (E&PDE16), Design Education: Collaboration and Cross-Disciplinarity, Aalborg, Denmark, 8th-9th September 2016*, pages 178–183, 2016.

A. Březina. Implementations of embedded systems in robotics systems (in slovak), pages 4-43. Master's thesis, DCAI, FEEI, Technical University of Košice, 2019.

M. G. Catalano, G. Grioli, M. Garabini, F. Bonomo, M. Mancini, N.s Tsagarakis, and A. Bicchi. Vsacubebot: A modular variable stiffness platform for multiple degrees of freedom robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 5090–5095. IEEE, 2011.

D. Cellucci, R. MacCurdy, H. Lipson, and S. Risi. 1d printing of recyclable robots. *IEEE Robotics and Automation Letters*, 2(4):1964–1971, 2017.

I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek. A brief introduction to opencv. In *2012 proceedings of the 35th international convention MIPRO*, pages 1725–1730. IEEE, 2012.

A. Jadlovská, S. Jadlovská, and D. Vošček. Cyber-physical system implementation into the distributed control system. *IFAC-PapersOnLine*, 49(25):31 – 36, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

J. Jadlovský and M. Kopčík. Distributed control system for mobile robots with differential drive. In *2016 Cybernetics & Informatics (K&I)*, pages 1–5. IEEE, 2016.

J. Jadlovský et al. Research activities of the center of modern control techniques and industrial informatics. In *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 279–285. IEEE, 2016.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

J. W. Romanishin, K. Gilpin, and D. Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4288–4295. IEEE, 2013.

J. M. Santos, D. Portugal, and R. P. Rocha. An evaluation of 2d slam techniques available in robot operating system. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2013.

M. Tkáčik. Embedded systems and their implementation in distributed control and monitoring systems (in slovak), pages 70-87. Master's thesis, DCAI, FEEI, Technical University of Košice, 2019.

M. Vanitha, M. Selvalakshmi, and R. Selvarasu. Monitoring and controlling of mobile robot via internet through raspberry pi board. In *2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, pages 462–466. IEEE, 2016.

J. Čerkala and A. Jadlovská. Application of neural models as controllers in mobile robot velocity control loop. *Journal of Electrical Engineering*, 68(1):39–46, 2017.

M. Yim, Y. Zhang, and D. Duff. Modular robots. *IEEE Spectrum*, 39(2):30–34, 2002.