A Low-Cost Real-Time Mobile Robot Platform (ArEduBot) to support Project-Based Learning in Robotics & Mechatronics

Ilya B. Gartseev[#1], Leng-Feng Lee[*2], and Venkat N. Krovi[*3]

[#]*Moscow State Technical University of Radioengineering, Electronics, and Automation, Moscow, Russia*
[1]gartseev@gmail.com

[*] *State University of New York at Buffalo, NY 14260 USA*
[2]llee3@buffalo.edu
[3]vkrovi@buffalo.edu

***Abstract*— We discuss aspects of development of a low-cost real-time mobile-robot platform – ArEduBot - for educational experiments. Our framework leverages ease-of- programming in block-diagrammatic form within the MATLAB/Simulink environment, together with several special-blocks developed within our Arduino-Simulink Toolbox [1] . The executable, compiled using the Real-Time-Workshop toolchain, can be downloaded for standalone real-time execution on an Arduino controller board interfaced to an iRobot Create mobile base. Our goal is to deploy this framework in introductory robotics and mechatronics classes, to complement the lecture and to support project-based learning. From this perspective, we compare the ease-of-use of multiple deployment architectures, describe our block-implementation within the Arduino Simulink Toolbox, and present several example experiments created using this framework.**

***Keywords*— project-based learning, mechatronics, robotics, arduino, simulink, matlab, educational platforms, mechatronics-enabled teaching and training; software design for system integration**

I. INTRODUCTION

Project-based learning encourages team-work, allowing students to learn from each other and preparing them for a real-work environments [1]. Our goal is to promote such project-based learning in various mechatronics/robotics courses (suitable for undergraduate seniors and incoming graduate students). To this end, we seek to create a flexible, open-ended, and easily programmable robotic kit to: (i) support students at various academic levels and with varying level of programming knowledge, (ii) support greater variety of algorithms and sensors, including the more complex ones in a deterministic real-time setting, while (iii) retaining the ease-of- programming in a high-level block-diagrammatic language.

Oftentimes, the selection of experimental test-platforms devolves to: (i) completely commercial-off-the-shelf (COTS) 'robotic kits'[2, 3]; (ii) 'robotic kit' designed completely by the instructor [4]; or (iii) using COTS components to assemble a customized robotic kit [5-7]. Examples of COTS robotic kits include Lego MindStorms by the Lego Group [8], iRobot products by iRobot Corporation [9], URTK robots by RosUchPribor company [10], Boe-Bot Kit by Parallax, Inc. [11], Make Controller Kit by Makezine [12] teamed up with MakingThings and Arduino-family boards [13]. However, they come with vast differences in terms of ease-of-use, cost, as well as underlying performance which tends to complicate selection. While completely COTS robotic kits facilitate immediate-use, often with considerable supporting tutorial and project materials, they tend to be expensive. In addition, such COTS kits focus more on the hardware-interfaces and thus often feature relatively simple/restricted software interfaces.

At the other end of the spectrum, developing a robotic kit completely in-house, takes significant investment of instructor-time –for kit design, kit maintenance (controller, chips, or software) as well as developing the supporting pedagogic material. Nevertheless, the instructor has the freedom to tailor the projects to align well with lecture material at a relatively low cost.

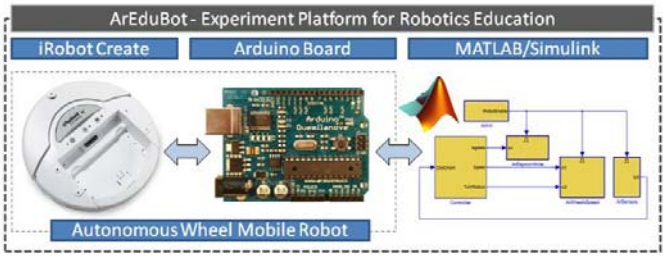


Figure 1: ArEduBot experiment platform consist of iRobot Create, Arduino Board, and MATLAB/ Simulink programming environment.

Hence, we sought an intermediary solution of building upon existing COTS components and adding to functionalities of basic robotic kit to create the ArEduBot. We chose the iRobot Create hardware (as a robust, inexpensive and well-supported platform), Arduino Board as our controller board (given its interrupt-based real-time capabilities, significant user- and code-base), and MATLAB/Simulink as the programming environment (given its wide usage for control courses). We propose to deploy the kit to support 3 robotics/controls courses: Mathematical Methods in Robotics

[1] The iRobot-Arduino-Simulink toolbox is available for downloading at http://www.gartseev.ru/projects/aredubot

(MAE 493/593) and Robotic Mobility and Manipulation (MAE 413/513) are senior/ graduate level course offered in the department of Mechanical & Aerospace Engineering at the University at Buffalo, and Automatic Control Systems in Mechatronics a undergraduate-senior level course offered in the department of Cybernetics in University 'MIREA' in Moscow, Russia. These three courses currently provide the theoretical flavor but lack a suitable easy-to-use platform for deployment. Most students came to the class with little or no prior knowledge in mechatronics, but are relatively comfortable with the MATLAB programming environment.

A. *Deterministic Real Time Operating Systems (RTOS)*

Deterministic real-time execution forms the bedrock of development of various estimation methods (e.g. velocity by finite differencing) or control methodologies (e.g. digital control). In the past, these requirements often restricted implementation to x86 or 68HC11 architecture systems that could facilitate low-jitter interrupt-based real-time code-execution. However, an often overlooked factor remains the ability to program such deterministic real-time algorithms within a user-friendly programming environment. Past deployments have required either direct assembly programming or at least C level programming. The auto-code-generation capability within a MATLAB/Simulink environment also placed minimal requirements for implementing interrupt based code – thereby effectively restricted deployment platforms to at least ARM-based processors (e.g. GumStix) or MPC555 (and their derivatives) .

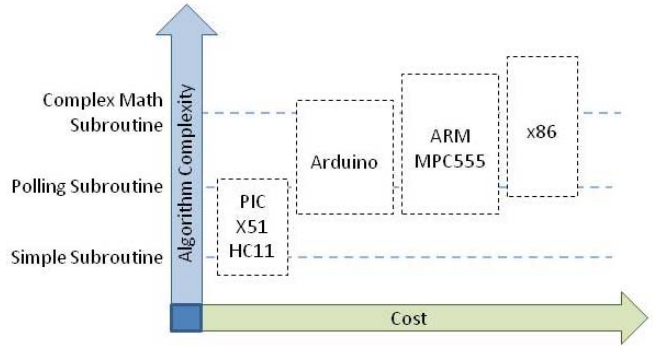


Figure 2: Using Arduino board fills the gap of handling complex algorithm at a relatively cost.

In recent times, the Arduino board has emerged as a viable processor-alternative to fill the gap between low-end microcontroller (PIC, X51, HC11) and the higher end processors (ARM, MPC555 and x86). However, there is a need for a framework to systematically develop deterministic real-time code. In this paper, we seek to remedy the absence of a suitable high-level block-library for the MATLAB Simulink, which can be used for block programming of the system.

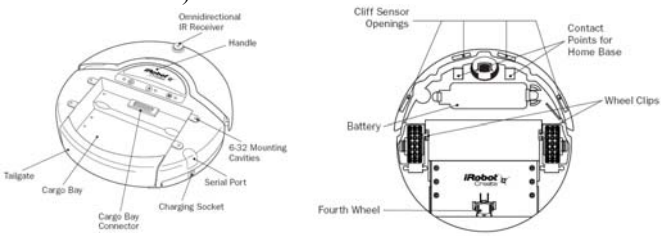
II. AREDUBOT COMPONENTS

It was crucial to develop an overall framework that allows students to easily test their algorithms on an inexpensive, real-time experimental testbed, within a relatively short period of time.

A. *iRobot Create*

The iRobot Create (shown in Figure 3) is a reprogrammable version of the Roomba robot vacuum cleaner for robotics hobbyists, educators, and researchers. This mobile platform is very actively used across the world to support research [14-16] and educational [17-19] activities in robotics and mechatronics.

The basic iRobot Create hardware, consists of two differentially-driven wheels, speakers, LEDs, cliff sensors, IR Receiver, serial port. iRobot Create features a command line API allowing for scripting from a command line interface via serial communication and can be programmed directly from PC. However, it is not always enough for the educational purposes and will be discussed further later in Section III. Nevertheless, this Create platform has been successfully employed to support a range of projects from easy (individually controlling wheels or reading sensors) to more complicated (trajectory planning with map construction and obstacle avoidance).



(a) **(b)**

Figure 3: (a) Top and (b) Bottom view of an iRobot Create Platform.

B. *Arduino Board*

The Arduino Board is a single-board open hardware design microcontroller, with embedded I/O support and a standard programming language [13]. While, there are variants of Arduino boards, we use the Arduino Duemilanove board [20] which retails for about $20. The ease of use of these boards is the reason of the wide spread usage for research [21, 22] as well as in education [23, 24]. Figure 4 depicts a sample Arduino board and contents of Arduino board kit while the core technical specifications are shown in Table 1.

(a) **(b)**

Figure 4: (a) Arduino Duemilanove board; and (b) A typical Arduino kit.

Arduino boards allows gathering information directly from digital sensors as well as by converting data from analog sensors using a built-in ADC chip. They also have the

capability and necessary hardware to drive different kind of motors with PWM outputs, including DC motors, servos, stepper-motors. Together with synchronous- and asynchronous-serial communication capability, this simplifies the implementation of closed-loop servo-control for a variety of mechatronic systems. The compact footprint promotes in-situ embedding of these devices in-situ, connected to a computer solely via a USB board interface. The Atmega328 microcontroller at the heart of the board can be re-programmed with AVR studio or any other compatible programmer with code developed in assembler and C language. Moreover, programs can be stored to EEPROM for the further autonomous device functioning. In addition, processor emulation capability with software and electrical circuits jointly (e.g. using Proteus [25]) offers an invaluable opportunity for the student projects.

TABLE I
SPECIFICATION FOR ARDUINO DUEMILANOVE BOARD

Microcontroller	ATmega328
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
Flash Memory	32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

C. MATLAB / Simulink

MATLAB/Simulink provide a user-friendly programming environment for students to develop code and subsequently download the compile executable for standalone operation to Arduino board. We focus on creating an Arduino Simulink Toolbox, which facilitates the above process. At this stage, we would also like to draw a clear distinction with respect to the MATLAB Toolbox for Create (MTIC) [26]. The MTIC provides a wrapper for Create Open Interface and this permits direct serial-communication from MATLAB to the iRobot Create. All code is developed, stored and executed on the host PC/base station. In contrast, our effort, we focus on developing standalone downloadable executables for Arduino board from within Simulink. This distinction will be clear in Section III, as we discuss some of the commonly encountered interaction architectures for the iRobot Create.

III. EXTANT & PROPOSED INTERACTION ARCHITECTURES

We discuss the various interaction-architectures for the iRobot Create and their potential advantages and disadvantages, within an educational setting.

A. Serial Port Communication (via API)

1) Terminal Program

The first option is connect the iRobot Create with PC through serial port, as shown on Figure 5. The solid circle denotes the hardware iRobot Create while the solid rectangles indicate software components, solid diamonds for hardware components, dashed rectangles for full-systems and dotted rectangles for functioning run-time system components.

The iRobot Create can be controlled from PC by sending of the numerical control sequences over a serial connection. Any computer with a serial interface may be used as terminal, e.g. realterm. This option is very useful for early-stage interactions and initial testing with iRobot but requires a wired run-time connection to the computer. While the limitation of wired connection can be overcome by using the Bluetooth Adapter Module (BAM) [27] or by using a RF based serial interface (e.g. Super Screamer), this increases the cost of the whole system significantly. Further, the interactive line-by-line interpreted mode of operation using an unintuitive and very limited serial script language is restrictive.

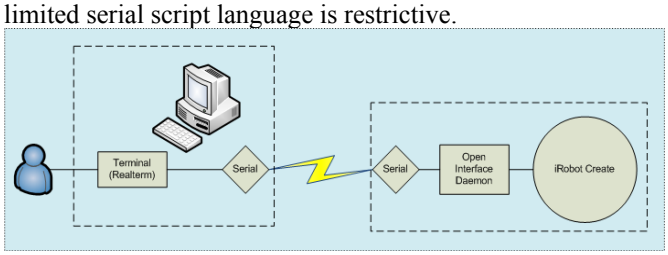


Figure 5: Control scheme for iRobot with terminal program

2) MATLAB/ serial port

The MATLAB Toolbox for the iRobot Create (MTIC) [26], shown in Figure 6, can be viewed as an extension of the above scripting process. It utilizes all of MATLAB computing power, to develop algorithms for iRobot Create. The toolbox function allow for translation of the high level commands (e.g. move forward) into the corresponding Open Toolkit API command directives (e.g. 134). In giving such wrapped function access to the iRobot Create it allows a programmer to write programs (instead of control with cryptic numerical sequences). However, from the hardware point of view, this option is equivalent to the PC terminal control, albeit with a slightly more capable software interface (MATLAB).

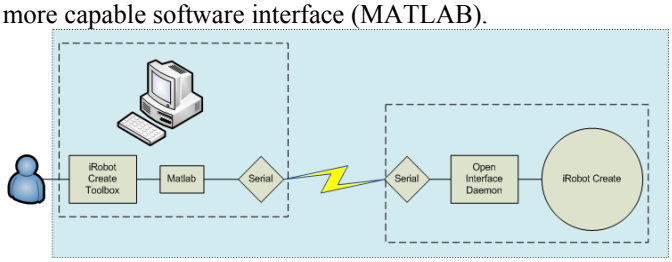


Figure 6: Control scheme for iRobot with MATLAB.

B. Dedicated Micro Controller

The Command Module Unit (CMU) for iRobot Create [28] is based on 8-bit, 18MHz Atmel Atmega 168 microcontroller, and can be powered by iRobot's battery. It enables full programmability of iRobot Create's motors, lights, sounds, and sensor sweeps and autonomous functioning. While permitting C programming, the CMU uses the Open Serial Interface to interact with the iRobot Create thus precluding direct control actuators or sensors. In Figure 7, the PC and CMU are connected via an USB interface for

programming the flash-memory after which the connection may be removed.

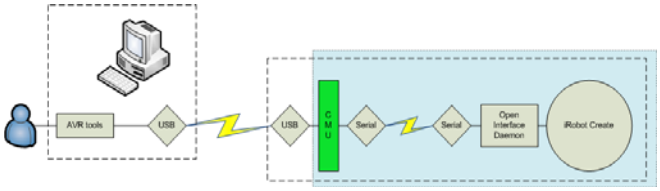


Figure 7: Control scheme for iRobot with CMU.

C. Embedded PCs

With embedded computers (x86 PC/104 system/ GumStix/ other ARM processor), it is possible to create a compiled executable using the MATLAB Real Time Workshop toolchain that can be downloaded to the memory of the embedded PC. The range of tasks that can be solved is very broad, by virtue of the greater computing power, but the cost is relatively high. The computational power permits more complicated and intensive computations (such as online trajectory planning). Moreover, most of the state-of-the-art embedded PCs, with wireless WiFi or Bluetooth interfaces, allowing the creation of wireless semi-autonomous/human-in-the-loop remote-controlled systems.

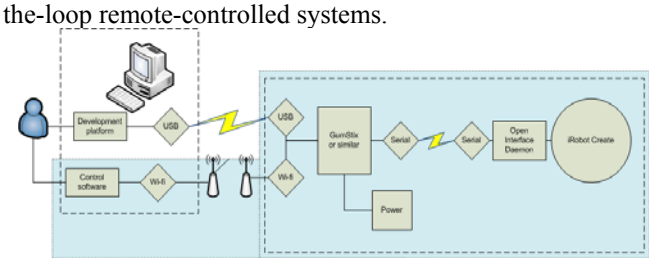


Figure 8: Control scheme for iRobot with embedded PC.

Figure 9 shows two ways of the running iRobot with microcontroller board. The short dotted rectangle depicts situation when iRobot and microcontroller are used in an autonomous mode. In this case, the development computer (high or low level) is used for compiling code and downloading onto the microcontroller board, while the system functioning in fully-autonomous mode at runtime. As a second alternative, highlighted by long narrow rectangle, the microcontroller board and controlling computer continue to use the Bluetooth or WiFi, interface as a data-logging /semi-autonomous control channel during runtime.

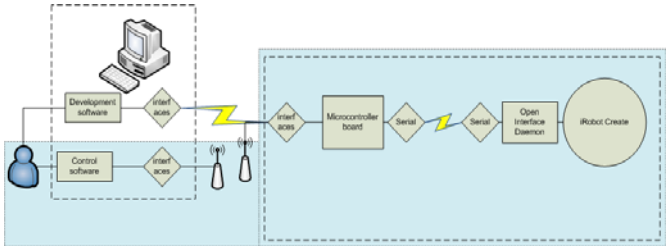


Figure 9: Control scheme for iRobot with microcontroller board.

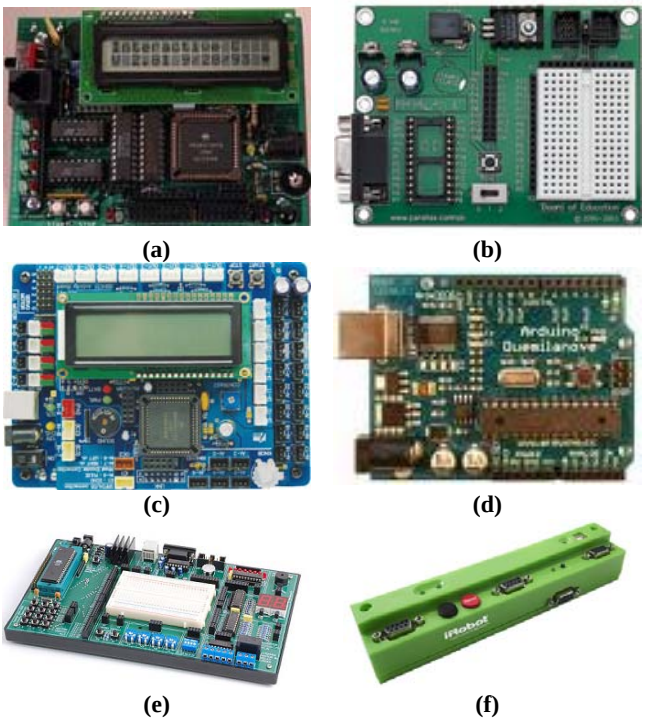


Figure 10: Microcontroller boards. (a) The Handy Board [29], (b) Parallax's Board of Education [11], (c) AX-11 Activity board [30], (d) Arduino Duemilanove [13], (e) SKIT-1RD2 V3 Experiment System [31], and (f) the Command Module Unit (CMU) from iRobot.

D. ArEduBot system

We take advantage of capability similar to that discussed in Section III.C, but with the Arduino Board mounted on the iRobot Create as shown on Figure 11. The Arduino Target tool [32] allows development of deterministic interrupt-based real-time code deployments for the Arduino platform right from Simulink. The target includes blocks to interface with the I/O ports on the Arduino board as well as a target file that automatically compiles and downloads the application onto the board directly from Simulink. We augment this process by developing a set of Level-2 S-function blocks encapsulated in a Simulink ArEduBot block library, discussed in Section IV. The MATLAB RTW toolchain can now be used to create executable machine language code directly from Simulink (unlike the MTIC [26] which also uses Simulink but only creates command level directives).

Figure 11: Hardware of ArEduBot system.

Figure 12 depicts logical structure of the ArEduBot system with three different ways of interacting with the ArEduBot: (i) low level programming; (ii) C level programming; or (iii) using the Arduino Target tool with our Simulink ArEduBot library for block-level programming of the Arduino board.

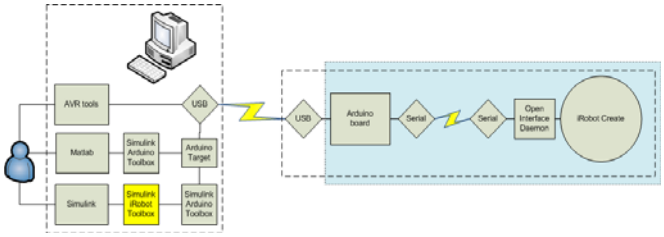


Figure 12: Control scheme for iRobot with Arduino board.

IV. Simulink ArEduBot Library

A control process for iRobot Create consists of interaction with different hardware parts of this device: wheels motors, sensors, sound source, LEDs, battery and, possibly, external devices. Consequently, the structure of the ArEduBot library should reflect control of all of these parts as shown in Figure 13, and library by itself should consist of blocks that realize all respective actions. Accordingly to aforementioned structure, a set of blocks in ArEduBot library is shown on Figure 14.

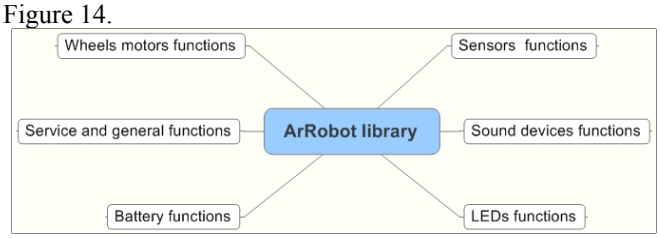


Figure 13: ArEduBot library structure.

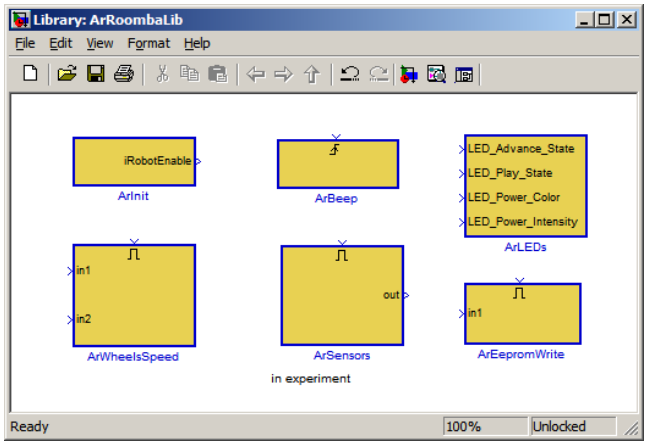


Figure 14: Blocks inside ArEduBot library.

A. ArInit block

This block initializes serial port for Arduino-iRobot connection and performs startup check of LEDs and beeper of iRobot Create. This block must be present in all models. The active level of output signal *iRobotEnable* of this block indicates that iRobot is initialized and ready to use.

This block has three parameters shown on Figure 15. *PauseBeforeInit* sets time before sending initializing sequence to the iRobot. *PauseAfterInit* sets time between sending initializing sequence to the iRobot and setting output signal to the active level. *InitType* sets one of initialization type: switching iRobot to the FULL operation mode only [33]; switching and lighting up LED indicators; switching, lighting up, and making a sound with beeper.

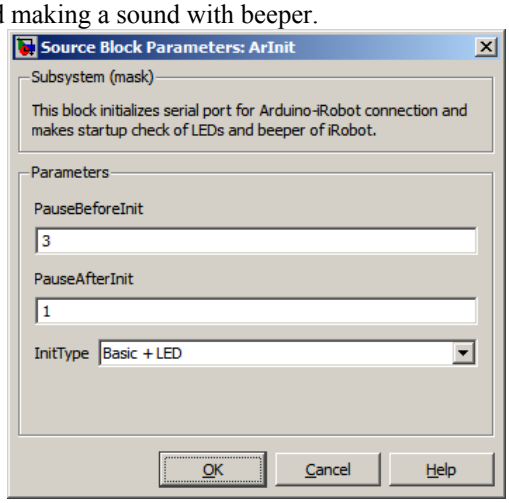


Figure 15: Parameters of ArInit block.

B. ArBeep block

This block initializes commands iRobot Create plays a sound that can be changed with block parameter *Song*. The parameter dialog is shown at Figure 16. Possible values for different sounds are discussed in [33].

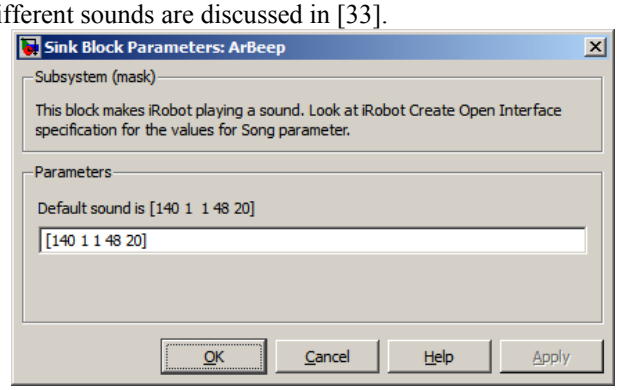


Figure 16: Parameters of ArBeep block.

C. ArLEDs block

This block controls state of the LEDs indicators on the iRobot Create. Input signals *LED_Advance_State* and *LED_Play_State* can be 0 (LED is off) or 1 (LED is on) only. Signals *LED_Power_Color* and *LED_Power_Intensity* can be in the range [0..255]. The value 0 of *LED_Power_Color* signal corresponds to green color of Power LED, 255 corresponds to the red color. In the same way, the value 0 of LED_Power_Intensity signal corresponds to minimal intensity of the LED, 255 corresponds to maximum level of the intensity. This block sends new value through serial connection only when one of the input signals is changed.

D. *ArWheelsSpeed block*

This block control the forward and backward motion of the ArEduBot wheels in various ways. The window with the block parameters is shown on Figure 17. The parameter *Mode* allows choosing between control options. The predefined value *'Direct wheels speed'* allows using of independent control values for both wheels in range of [-500 mm/s; 500 mm/s]. With choosing *'Angular velocity'*, it becomes possible to use angular speed in rad/s as inputs of this block. Last mode *'Forward velocity and turn radius'* allows the control of both linear and turn motion of ArEduBot simultaneously. First input of the block sets linear speed in mm/s. Second input sets turn speed of the robot in rad/s.

This block sends new values through serial connection only when one of the input signals is changed and when Enable signal for the block has active level (more than 0). Commonly, Enable input should be connected to the Enable signal, generated by ArInit block; however, it can be used in ways that are more sophisticated.

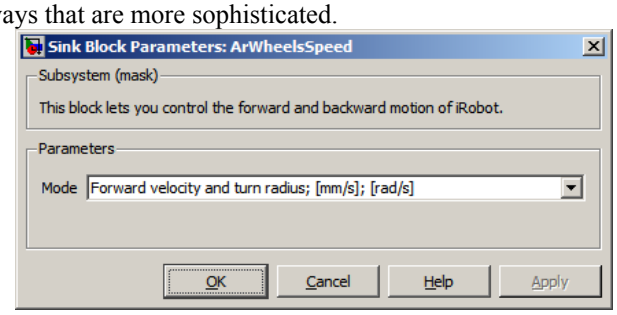


Figure 17: Parameters of ArWheelsSpeed block.

E. *ArSensors block*

This block creates an interface to the sensors of the iRobot Create. By means of parameter *Sensor Packet* user can get information from a sensor of interest. The value of this parameter is integer number in the range from 0 to 42 and it corresponds to the sensors packages predefined by iRobot Open Interface documentation. For example, if Sensor Packet parameter has value of 39, output signal of block will have vector with two values of current velocity of left and right wheels.

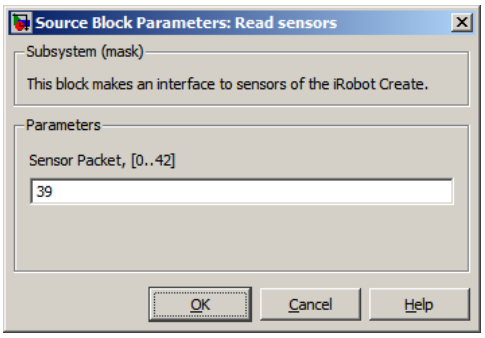


Figure 18: Parameters of ArSensors block.

F. *ArEepromWrite block*

This block writes input values to the EEPROM memory of Arduino board sequentially. The input signal should have the type of 8-bit unsigned integer. The meaning of input values can be any. The block has input Enable, which allows writing when stat is active.

V. Application Example

The combination of the above blocks allows one to create basic operations on the iRobot Crete through MATLAB/Simulink interface. Two such operations are shown here.

A. *Case 1: Moving forward and backward*

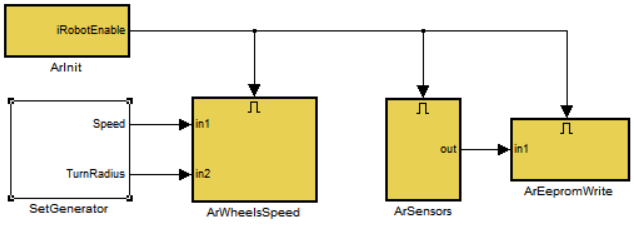


Figure 19: Application of ArEduBot library: moving forward and backward

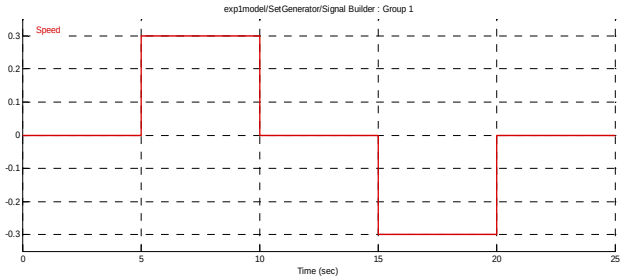


Figure 20: Desire speed input to the iRobot to move forward and backward.

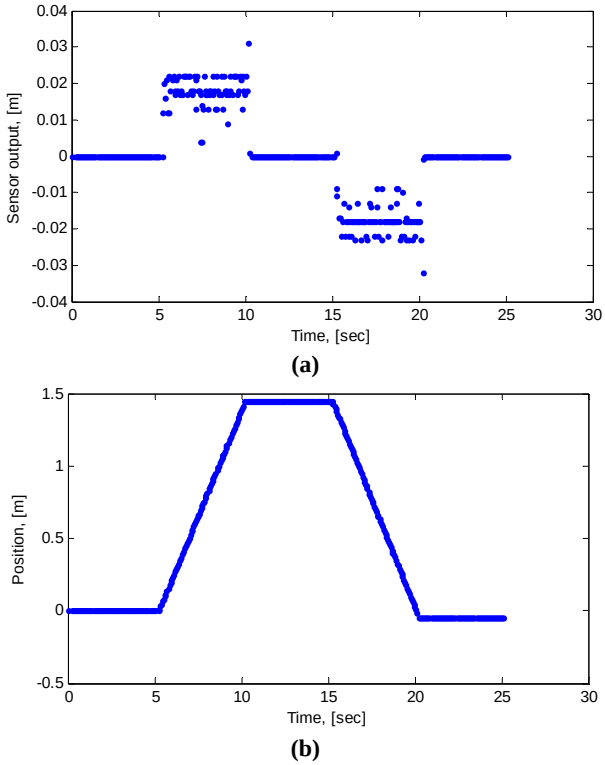


Figure 21: (a) Position differences output from the sensor; and (b) the position of the iRobot traveled by integrating the position difference in (a).

The typical application with ArEduBot library is shown on Figure 19. The block ArInit runs before anything else and generates active level of its output signal after successful initialization of the ArEduBot system. The block ArWheelsSpeed is working in the mode 'Forward velocity and turn radius'. The model gets desired speed and turn radius of iRobot from the block SetGenerator. Generally, they can be simply constants or more complicated time-dependent signals. Block ArSensor provides value of distance difference and orientation difference between current time step and previous time step. These values packed into two-components vector. Block ArEepromWrite stores values to the Arduino's EEPROM.

The input signal to the iRobot is shown in Figure 20, where we provide a forward speed input of 0.3m/s from time = 5 sec to 10 sec. Stop for 5 sec, and move backward at 0.3m/s for another 5 sec. Figure 21(a) shows the position difference reading from the iRobot sensor, and by integrate these sensor reading, one can obtain the position traveled by the iRobot shown in Figure 21(a).

B. Case 2: Turning

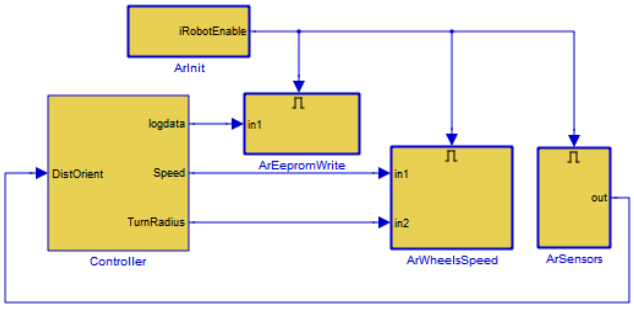


Figure 22: Parameters of ArSensors block.

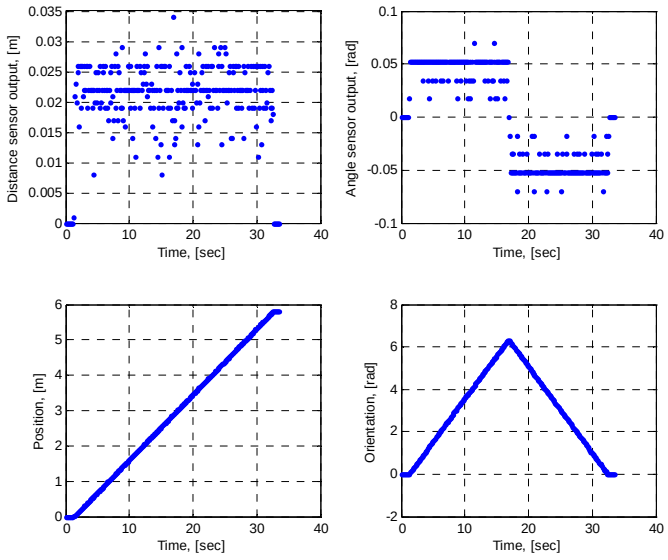


Figure 23: Position (top left) and angular (top right) differences output from the sensor; and the position (lower left) and angle (lower right) of the iRobot by integrating the position difference.

This example demonstrates a simple close loop control for the iRobot. In this case, the desired trajectory is a "∞" trajectory. The block Controller controls output signals to make iRobot perform motion along the two conjugate circles. The feedback from the sensors allows switching from the one circle to another. The signals from sensors and calculated robot' position and orientation are shown on Figure 23. The robot' trajectory is shown on Figure 24.

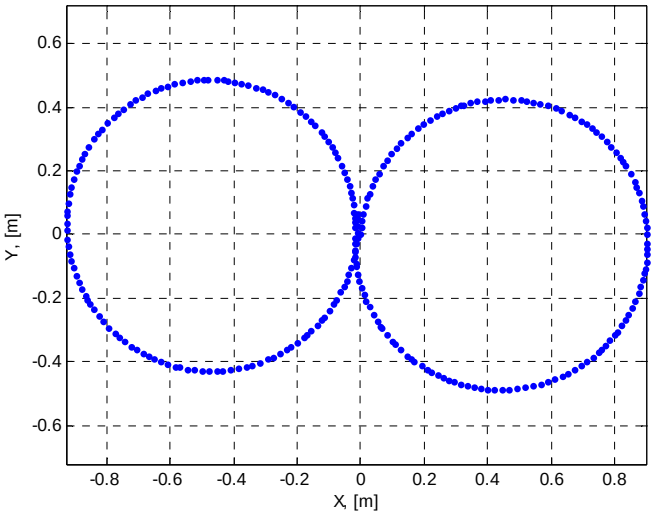


Figure 24: The robot traveled trajectory.

VI. CONCLUSION

We presented a Simulink Toolbox for ArEduBot, that uses iRobot Create as the mobile platform, Arduino board as the controller, and MATLAB/Simulink as the programming interface, to serve as an experimental platform for robotic courses. The ArEduBot has the advantage of the ability to utilize many of the MATLAB/Simulink blocks and its convenient programming environment to create executables for standalone deterministic real-time execution on the Arduino board interfaced with the iRobot Create, at relatively low cost. This Toolbox is intended for use in introductory robotics class where student have limited background in mechatronics but want to create a robotic system to implement what they have learned in lecture. We are in the process of deploying this Toolbox in multiple robotic classes to support the associated group projects.

REFERENCES

[1] J. Chizmar and D. Williams, "What Do Faculty Want?," *EDUCAUSE Quarterly*, vol. 24, pp. 18-24, 2001.

[2] F. Klassner, "A Case Study of LEGO Mindstorms'™ Suitability for Artificial Intelligence and Robotics Courses at the College Level," in *33rd SIGCSE technical symposium on Computer science education*, Cincinnati, Kentucky 2002.

[3] S. Parsons and E. Sklar, "Teaching AI using LEGO Mindstorms," in *AAAI Spring Sympos on Accessible Hands-on Artificial Intelligence and Robotics Education*, 2004.

[4] J. Fiene. (2010). *The MaEvArM Project*. Available: http://alliance.seas.upenn.edu/~medesign/wiki/index.php/Guides/MaEvArM

[5] I. B. Gartseev and E. N. Kryuchenkov, "Hardware Development for Autonomous Mobile Robot," in *State-of-the-Art Information Technologies*, Sudak, Ukraine, 2006.
[6] I. B. Gartseev, *et al.*, *The Training Module Robot UMR-2: Datasheet and Manual*. Moscow: RosUchPribor-MIREA, 2002.
[7] I. B. Gartseev, *et al.*, "Unified Control System for Mechatronic Modular Robots," in *State-of-the-art Technologies in Control, Automatics, and Computer Science*, Ukraine, 2003.
[8] LEGO. (2010). *Lego MindStorms*. Available: http://www.lego.com/eng/info/
[9] iRobot. (2010). *iRobot Create*. Available: http://www.irobot.com/
[10] RosUchPribor. *URTK Robots*. Available: http://www.rosuchpribor.ru/
[11] Parallax. *Boe-Bot Kit*. Available: http://www.parallax.com/
[12] O'Reilly. *The MAKE team*. Available: http://makezine.com/about/
[13] Wiki-resource. *Arduino*. Available: http://www.arduino.cc/
[14] D. Smith and Z. Dodds, "Visual navigation: Image profiles for odometry and control," ed, 2009, pp. 1188-1189.
[15] N. Correll, *et al.*, "Ad-hoc wireless network coverage with networked robots that cannot localize," ed, 2009, p. 3878.
[16] Z. Koziol, *et al.*, "A vision for spatial-reasoning commodity robots," ed, 2009, p. 195.
[17] Z. Dodds and B. Tribelhorn, "Erdos: cost-effective peripheral robotics for AI education," ed, 2007, pp. 1966-7.
[18] P. Mawhorter, *et al.*, "A Tale of Two Platforms: Low-Cost Robotics in the CS Curriculum," vol. 24, ed, 2009, pp. 180-188.
[19] I. O. Richard Weiss, "Finding your bot-mate: criteria for evaluating robot kits for use in undergraduate computer science education," vol. 24, ed, 2008.
[20] D. Jolliffe. (2006). *Arduino Fever*. Available: http://cdn.makezine.com/make/arduinoMAKE07.pdf
[21] B. Batagelj, *et al.*, "Digital airbrush," in *ELMAR-2009 - 51st International Symposium ELMAR-2009, September 28, 2009 - September 30, 2009*, 2009, pp. 306-308.
[22] D. Bri, *et al.*, "A multisensor proposal for wireless sensor networks," presented at the 2nd International Conference on Sensor Technologies and Applications, SENSORCOMM 2008, Cap Esterel, France, 2008.
[23] J. D. Brock, *et al.*, "Using Arduino for Introductory Programming Courses," vol. 25, ed, 2009.
[24] H. Blemings and J. Oxer, *Practical Arduino: Cool Projects for Open Source Hardware*, 1 ed.: Apress, 2009.
[25] L. E. Ltd. *Proteus PCB Design Software*. Available: http://www.labcenter.co.uk/index.cfm
[26] J. M. Esposito, *et al.* (2008). *Matlab Toolbox for the Create Robot*. Available: www.usna.edu/Users/weapsys/esposito/roomba.matlab/
[27] I. Element Products. (2007). *BAM for the iRobot® Create™. User Manual*. Available: http://www.elementdirect.com/files/10542B.pdf
[28] i. Corporation. (2007). *iRobot® Command Module. Owners manual*. Available: http://www.irobot.com/filelibrary/create/Command%20Module%20Manual_v2.pdf
[29] *The Handy Board*. Available: http://handyboard.com/hb/
[30] V. R. Kit. (2008). *VegaROBOKit*. Available: http://vegarobokit.com/
[31] P. Limited. (2009). *SKIT-51RD2 V3 Experiment System*. Available: http://www.paltronix.com/skit-51rd2v3.htm
[32] D. Eastman. (2009). *Arduino Target*. Available: http://www.mathworks.com/matlabcentral/fileexchange/24675
[33] iRobot. (2006). *iRobot Create Open Interface Specification*. Available: http://www.irobot.com/