IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 30, 2017, accepted May 27, 2017, date of publication June 13, 2017, date of current version July 3, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2715184

A Light-Weight Rapid Control Prototyping System Based on Open Source Hardware

YOUNG SAM LEE[1], BONGEON JO[1], AND SOOHEE HAN[2], (Senior Member, IEEE)
[1]Department of Electrical Engineering, Inha University, Incheon 22212, South Korea
[2]Pohang University of Science and Technology, Pohang 37673, South Korea

Corresponding author: Young Sam Lee (lys@inha.ac.kr)

This work was supported by Inha University Research Grant.

ABSTRACT In this paper, we propose a light-weight rapid control prototyping (RCP) system based on off-the-shelf open source hardware to achieve high performance computing, cost effectiveness, portability, and easy accessiblity. The proposed RCP system consists of a PC-based computer-aided control system design (CACSD) tool for computing control action and tiny palm-sized open source hardware for input and output (I/O) operation and data transfer through a built-in high-speed USB interface. The popular-priced and portable open source hardware is used as a bridge between CACSD tools and real plants to deliver the control and sensor data at the sampling time points. Ten I/O function blocks written in C code are developed based on the CACSD tool employed for the proposed light-weight RCP system in order to handle I/O operation in a simple way. In addition, we suggest two practical strategies, a batch transfer strategy and a variable sampling period method, to increase the sampling rate of the control system. It is shown through experiment that the proposed light-weight RCP system works well up to a sampling rate of 2 kHz without adopting expensive hardware and C code generators. It is expected that the proposed RCP system will be considered as affordable and readily available to schools for mass education.

INDEX TERMS Arduino due, rapid control prototyping (RCP), block libraries, high-speed USB interface.

I. INTRODUCTION

As an efficient control design method, Rapid Control Prototyping (RCP) has been widely used in many engineering fields such as mechatronics [1], [2], automobiles [3], motion control [4], and so on. RCP provides a fast and cost-effective way for control engineers to quickly create working control system prototypes, verify their control algorithms on hardware in similar environments to the real ones, and evaluate their control performances before entering the embedded implementation stage. Such early verification through RCP enables control engineers to easily adjust their designs until they are satisfied with the results, allowing them to have confidence that their designs will work in the field. In other words, RCP reduces the development time and effort by automating many of the demanding and repetitive tasks and by allowing corrections and changes to be made early when they are easy. In this regard, an RCP-based control design can make control engineers focus on their main tasks instead of tedious tasks for developing infrastructure such as hardware interfaces, data acquisition and transmission, and data plotting.

Existing RCP systems utilize target hardware for control computation. Target hardware can be a computer or an embedded microcontroller board. If the target is a computer, I/O boards are required for I/O operation. Code generation software tools automatically generate target-specific C codes from block diagrams of the control algorithms. Executable programs made from generated C codes run on the target hardware in a real time. A good survey on the existing commercial RCP systems is given in [5]. Comparisons among those RCP tools are also given therein. Such commercial RCP systems show good performance in terms of their achievable sample rate, real-time property, and diversity in I/O functions. However, they are mostly very expensive and overwhelmed with unnecessarily high and numerous advanced options. Furthermore, I/O boards should be fixed to a particular computer because their interfaces are PCI, ISA, or PCI Express. For this reason, existing RCP systems are not portable and are hence inconvenient when they need to move frequently.

In addition to commercial RCP systems, it has been reported that laboratory-level RCP systems were also developed to design controls with low costs and moderate

performance [6]–[9]. However, these non-commercial RCPs are difficult to popularize due to issues with their reliability and standardization. Even though they are not commercialized, they also require a commercial C code generator, which implies that they are not free from financial issues.

RCP systems for educational purpose should have cost effectiveness and portability and should be based on readily available hardware. However, existing RCP systems do not satisfy all of these requirements. These issues have motivated the research presented in this paper.

Recently, open source hardware has become prevalent since its performance is great compared with its price and its reliability is guaranteed by securing many users [10]–[12]. According to the definition given in [13], open source hardware is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design without paying royalties or fees. Since it comes with a variety of performances, sizes, and price ranges, we can develop our own applications with proper performance and an affordable price. Open source hardware gives us the freedom to realize our own creative ideas with an easy development environment and plug-in interfaces. Such useful and efficient open source hardware has been used for rapid prototyping of image processing [14], communication systems [15], manufacturing automation [16], [17], scientific instrumentation and research [18], and so on.

Recently, MATLAB/Simulink, which is one of the most well-known CACSD tools, has begun supporting some RCP systems using open source hardware, which shows that even big companies now realize the potential of open source hardware. In this respect, it would be very meaningful and practical to make use of open source hardware for implementing RCP methods.

In this paper, we propose a light-weight RCP system based on the aforementioned open source hardware in order to achieve high performance computing, cost effectiveness, and portability. The proposed RCP system consists of a PC based computer-aided control system design (CACSD) tool for computing the control action and tiny palm-sized open source hardware for I/O operation and data transfer through built-in high-speed USB interfaces. Since a PC has superior computing power compared with embedded processors, complicated computations such as population-based optimization and nonlinear programming can be efficiently performed. Adopting PC-based computing instead of embedded computing, we can provide high performance with only a small sacrifice in communication latencies. Replacing expensive and burdensome embedded processors and I/O boards, popular-priced and portable open source hardware is used as a bridge via high-speed USB interfaces between CACSD tools and real plants to deliver control and sensor data at the sampling time points. To connect controls to real plants in a visual and easy way, ten I/O function blocks written in C code are developed based on the CACSD tool employed for the proposed light-weight RCP system, which sends the control data to or receives the sensor data from the plant. In addition, we suggest two practical strategies, a batch transfer strategy and a variable sampling period method. The batch transfer strategy can reduce the overheads required by each loop, and then it drastically improves the communication efficiency. The variable sampling period method provides more correct control by making good use of the time information retrieved from the open source hardware. It is shown through experiment that the proposed light-weight RCP system works well up to the sampling rate of 2 KHz without adopting expensive hardware and automatic C code generators. It is expected that the proposed RCP system will be considered as affordable and easily available to schools for mass education. Furthermore, the proposed system is believed to contribute to the growing popularity of RCP-based control design.

The remainder of this paper is organized as follows. In Section II, an overview of the proposed light-weight RCP is given together with its basic components. Software for the proposed RCP system will be described in Section III. An example of controller implementation and a supplemental explanation using this example will be provided in Section IV. A comparison with existing RCP systems is made in Section V. Finally, conclusions are drawn in Section VI.

II. THE PROPOSED LIGHT-WEIGHT RCP SYSTEM

There exist several CACSD tools such as MATLAB/Simulink [19], Labview [20], Scilab/Scicos [21], and CEMTool/SIMTool [22]. All of them provide graphical programming methods. Among those CACSD tools, MATLAB/Simulink may be the most widely used because of its powerful functionalities and easy interface. In this paper, we choose to use MATLAB/Simulink as the CACSD tool in which to implement the proposed RCP system. It is mentioned that the proposed RCP system can also be implemented into other CACSD tools through simple customization.

A. AN OVERVIEW OF THE PROPOSED RCP SYSTEM

The schematic diagram given in Fig. 1 shows how the proposed RCP system works. It shows that the proposed RCP system consists of two subsystems, excluding the plant system. The first subsystem is a PC system, in which Simulink is running under MS Windows. The second subsystem is an Arduino Due board, which has a built-in high-speed USB interface.

The PC performs control computations and the Arduino Due is in charge of I/O operation through its peripherals such as ADC, encoder counter, DAC, and PWM. The PC runs a Simulink controller model in which the sensor data sent from the Arduino Due are received through input blocks supported by the proposed RCP system, the control values are computed using the received sensor data, and the computed control data are then sent to the Arduino Due using the output blocks, which are also supported by the proposed RCP system. Data communication between the PC and the Arduino

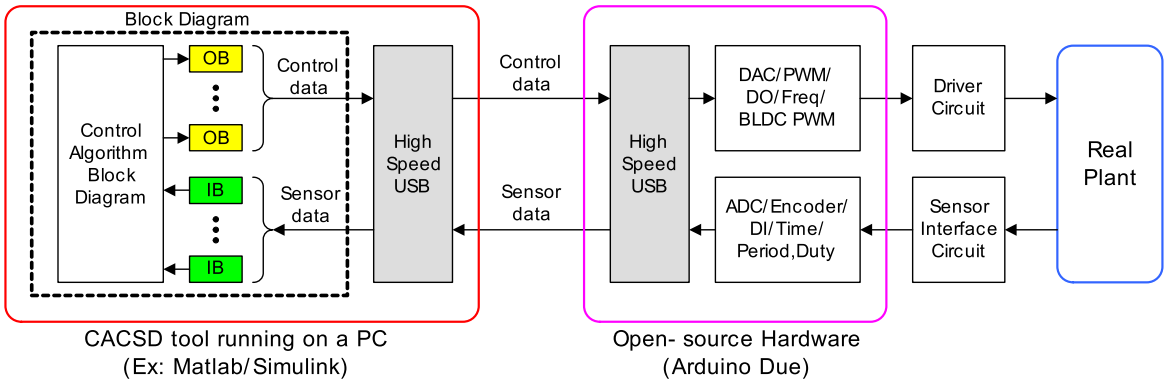

FIGURE 1. The schematic diagram of the proposed RCP system (IB: Input block, OB: Output block).

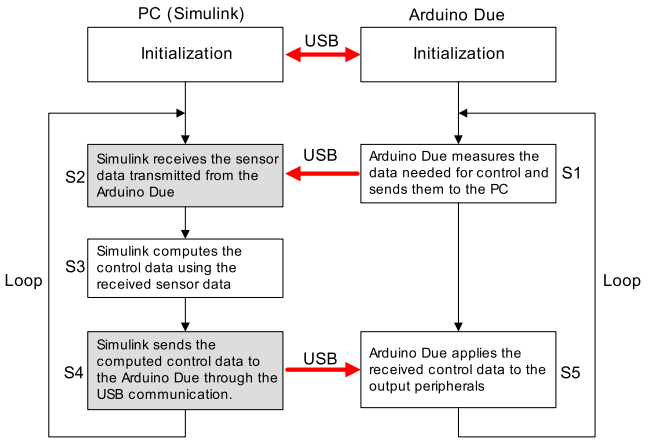

FIGURE 2. Flowchart of the proposed RCP system.

Due is done through high-speed USB communication in order to minimize the latency. The kinds of sensors and control data to be received from and transmitted to the Arduino Due are determined by the I/O blocks provided by the proposed RCP system in a controller model constructed using Simulink.

The controller model is built using various built-in blocks provided by Simulink and the I/O blocks supported by the proposed RCP system. The control action is further described using the flowchart in Fig. 2. S1 and S5 are performed in the Arduino Due and S2, S3, and S4 are performed on the PC. For periodic real-time control operation, we utilize the SysTick timer of the Arduino Due instead of using a real-time kernel from the PC. Even though the sample time has a small amount of jitter, it is overall real-time, which will be demonstrated later in this paper.

In most existing RCP systems, the controller model constructed from Simulink is translated into C code by an automatic code generator, compiled, and then downloaded to the target hardware, which performs real-time control operations. On the other hand, the proposed RCP system does not go through the code generation process. Instead, the Simulink controller model itself acts as a real-time controller. Because the proposed RCP system does not need cumbersome code generation, change and verification of the controller can be done quickly. It is mentioned that the code generator is an expensive tool, which costs several thousand dollars and is not contained in a basic version of MATLAB, but instead provided as a specialized expensive toolbox. Furthermore, compilers are not required either.

The reason why we choose the structure mentioned above is because we want the proposed RCP system to be low-cost, portable, and easily accessible with only a moderate sacrifice in the fast sample rate of the control loop. Low cost stems from the two facts: no additional software other than MATLAB/Simulink is required, and the required hardware is very cheap. Low cost and portability is further explained in the next subsection.

B. ARDUINO DUE AS AN I/O BOARD

In this paper, we use the Arduino Due, which is a well-known open source hardware platform, as an I/O board. The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital I/O pins of which 12 can be used as PWM outputs, 12 analog inputs, 2 DAC, a 84 MHz clock, a built-in native USB port, etc. Fig. 3 shows a picture of the Arduino Due board. The reasons why we choose the Arduino Due as an I/O board are as follows:

- It is an open source hardware platform and hence well-known, widely used, easily accessible, and affordable. Its price is less than 50 dollars.
- The size of the board, 101.6 mm$\times$53.3 mm, is small enough to carry.
- It is based on a microcontroller adopting ARM Cortex-M3 core, which has a Nested Vectored Interrupt Controller (NVIC).
- It has plentiful I/O peripherals that can be used for control purposes.
- It provides a built-in high-speed USB interface that can transfer data at a fast speed.

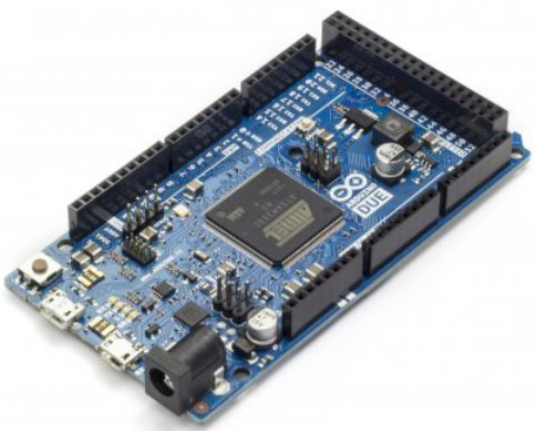

FIGURE 3. Arduino Due : a microcontroller board.

- A free IDE (integrated development environment), such as Atmel Studio, is available.

In the proposed RCP system, the Arduino Due is not in charge of the control computations. Instead, it only measures the necessary sensor data and then sends them to the PC so that Simulink can compute the required control value based on the sensor data. The computed control data are sent back to the Arduino Due and then applied through output peripherals. The Arduino Due is widely used and its price is less than 50 dollars. In the proposed RCP system, the actual computation of the control is performed on the PC side. Thus, we do not need such a high-speed microcontroller. The Arduino Due has a built-in high-speed USB interface. As is widely known, USB supports plug-and-play and can be easily interfaced with any PC, including both laptops and desktops. Hence, the USB interface contributes to the good portability of the proposed RCP system. On the other hand, the I/O boards supporting interfaces such as PCI, ISA, or express PCI are installed to a PC after the power-off and installation change of the board to another PC cause some inconvenience, which is not the case in the proposed RCP system.

Nowadays, engineering education has put much emphasis on hands-on experiments and design. RCP-based mass education of engineering undergraduates requires that the RCP system be low-cost. Since support for take-home experiments or design would enhance students' understanding, portability of the RCP system is also a prerequisite. Therefore, the Arduino Due, which satisfies both cost effectiveness and portability, may be the best choice for the I/O board for the proposed RCP system.

The proposed RCP system provides ten I/O blocks that are used to control some important I/O peripherals. By using those blocks, transfer of the data to and from the Arduino Due can be easily achieved.

C. RCP I/O BLOCKS

The most important feature of RCP systems is that they provide I/O blocks so that controller designers can easily handle I/O operation. Thus controller designers do not have to spend much time struggling with handling peripherals. Instead, they have only to focus on the control algorithm itself with the help of the provided I/O blocks. The proposed RCP system provides ten I/O blocks so that the controller can handle I/O operation using the Arduino Due within the Simulink environment. TABLE 1 lists the supported I/O blocks and the corresponding descriptions. Fig. 4 shows those I/O blocks that can be used with Simulink. The blocks in the left column are input blocks and the blocks in the right column are output blocks. Some blocks support multiple channels. For example, the ADC block has 8 channels. In order to change the channel configuration, we can double-click the block and then choose the channel number through a dialog box. Fig. 5 shows the dialog box of the ADC block.

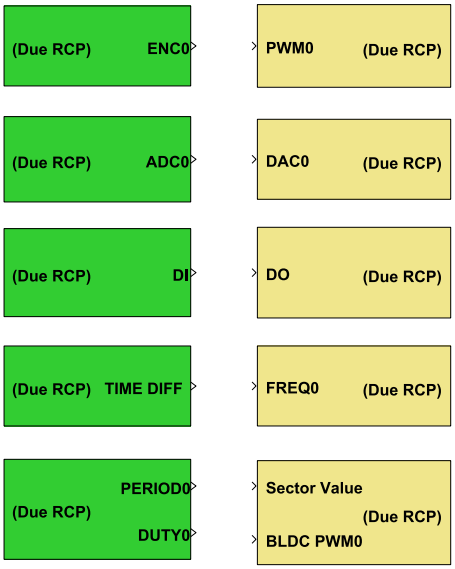

FIGURE 4. I/O blocks supported by the proposed RCP system.

TABLE 1. Descriptions of the I/O blocks supported by the proposed RCP system.

	Block name	Description
Input blocks	Encoder counter	2 channels, 32–bit
	ADC	8 channels, 12–bit
	Digital input	8–bit
	Period and duty	3 channels
	Time/Time difference	resolution : micro second
Output blocks	PWM	6 channels PWM/Direction interface or Unipolar complementary PWM
	DAC	2 channels, 12–bit
	Digital output	8–bit
	Frequency	2 channels, up to 40,000 Hz
	BLDC PWM	2 channels

Among those ten I/O blocks, the PWM block and Time block will be more detailed as follows:

- PWM block: PWM is usually used for controlling electric motors, and the PWM block supported by the proposed RCP system provides two types of PWM

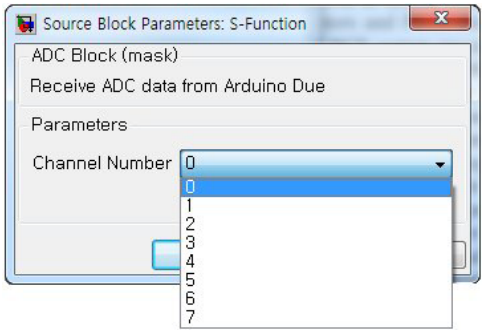

FIGURE 5. Channel selection of the ADC block through a dialog box.

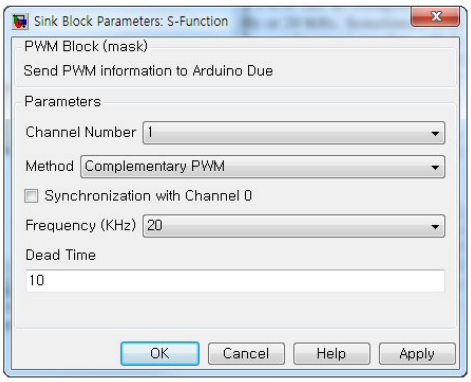

FIGURE 6. Dialog box of the PWM block.

interface, the PWM/Dir interface and complementary PWM interface. The frequency of PWM can be configured. PWM signals are sometimes required to be synchronized together, which means that those PWM signals are all generated from a single counter. For this purpose, the PWM block provides options for synchronization. Furthermore, the dead time for each PWM channel can be specified independently. Fig. 6 shows the dialog box of PWM block by which all options mentioned above can be specified.

- Time block: The Time block can be used to obtain the elapsed time from the Arduino Due. Two types of time information can be obtained: time and time difference. Time is the system time itself and time difference is the difference in the system time between the previous sample and the current sample. Fig. 7 shows the dialog box for the Time block.

D. A PIN MAP FOR THE PROPOSED RCP SYSTEM

The proposed RCP system supports ten I/O blocks. Each I/O block has its own pins on the Arduino Due. Fig. 8 shows the pin map diagram developed to help users find the pins dedicated to the I/O blocks supported by the proposed RCP system. In order to use a particular I/O operation, we can check the position of the pins for the chosen I/O operation by referring to the pin map and connecting the appropriate wire. We then put the corresponding I/O block in the

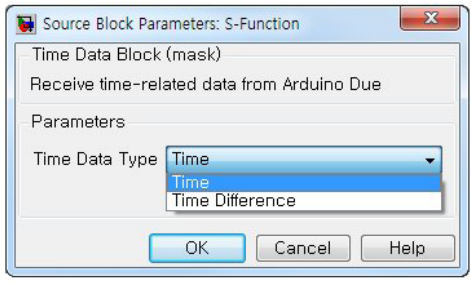

FIGURE 7. Dialog box of the Time block.

Simulink model. By running the completed Simulink model, we can easily handle the required I/O operation.

III. SOFTWARE OF THE PROPOSED RCP SYSTEM

In this section, we describe the software programs that have been developed to implement the proposed RCP system. One program runs on the Arduino Due and the other program runs on a PC within the Simulink environment. The Simulink-side program has been developed through an S-function written in C code. S-functions provide a powerful mechanism for extending the capabilities of the Simulink environment. An S-function is a computer language description of a Simulink block written in MATLAB, C, C++, or Fortran [23]. Users can write anS-function to describe a user-defined function block. In order to handle the USB communication and improve the computation speed, we developed an S-function in C. The program for the Arduino Due has been developed in C under Atmel Studio [24].

Two programs interact with each other as shown in the flow chart in Fig. 2. It illustrates that I/O operations are taken care of by the Arduino Due and the control computation is performed by Simulink running on the PC. Before the loops on both sides begin, some necessary initialization steps for each side take place. During the initialization phase, the S-function program determines the type of I/O blocks used in a Simulink model, the execution order among those I/O blocks, the sample time of the Simulink model, etc., and sends the gathered information to the Arudino Due. During the initialization phase, the program on the Arduino Due receives information on I/O blocks from Simulink, appropriately initializes the peripherals corresponding to those I/O blocks, and notifies the Simulink that the Arduino Due is ready to start a loop. The flow chart shown in Fig. 9 illustrates this initialization phase for both sides sequentially.

Once the loops begin on both sides, procedures shown in the flow chart in Fig. 10 will be repeated until the end of simulation. During the loops, the data transmission between two sides is performed on the basis of batch transfer strategy, which will be more clarified in the next section. Further explanation of the flow chart is given as follows:

- The Arduino Due reads the sensor data using the input peripherals and fills out TxData[] with the acquired data. The kind of input peripherals and the order in which they are read are determined from the information

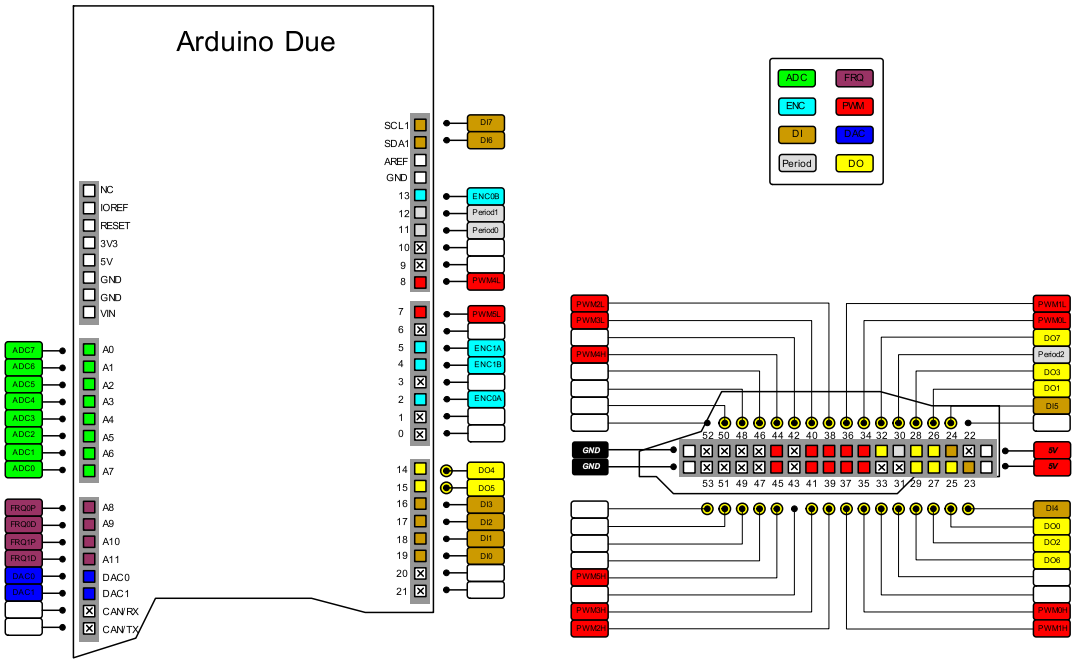

FIGURE 8. Pin map on the Arduino Due developed for the proposed RCP system.

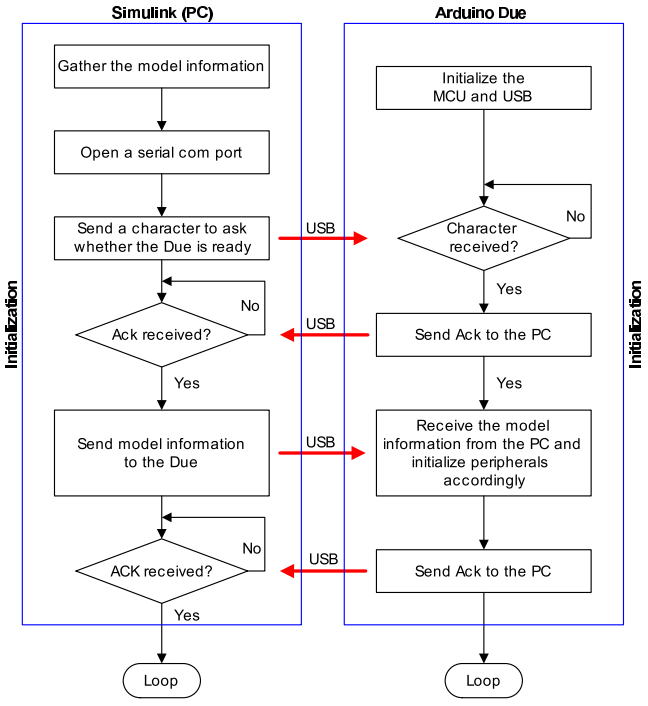

FIGURE 9. Flowcharts for the initialization phase.

(KindOfMeasure[]) received from Simulink during the initialization phase. After all the acquired data are filled in TxData[], they are batch-transferred to the PC through USB communication.

- The I/O block that has the first execution order among all I/O blocks in the Simulink model receives the batch-transferred data from the Arduino Due and store them in RxBuffer[].
- If any block is an input block, it retrieves its data from RxBuffer[] and update the output of the input block using the retrieved data. Offset information on that block in RxBuffer[] is utilized to obtain the correct data from RxBuffer[].
- If any block is an output block, it processes its input data appropriately and fills them in TxBuffer[] at the proper position. Offset information on that block in TxBuffer[] is utilized to get the correct position in TxBuffer[].
- The I/O block that has the last execution order among all I/O blocks in the Simulink model batch-transfers all the data in TxBuffer[] filled by output blocks to the Arduino Due through USB communication.
- The Arduino Due receives the batch-transferred control data from the PC and stores them in RxData[]. The control data are then retrieved and applied to the corresponding output peripherals. The kind of output peripherals and the order in which they are accessed are determined from the information (KindOfControl[]) received from Simulink during the initialization phase.

The control system is implemented such that it performs periodic control operation according to a constant sample time. In the proposed RCP system, we use the SysTick timer of the Arduino Due in order to determine the elapsed time. The elapsed time in micro-seconds can be determined from the timer and enables periodic control operation. The elapsed time information is available to the control algorithm using the Time block.

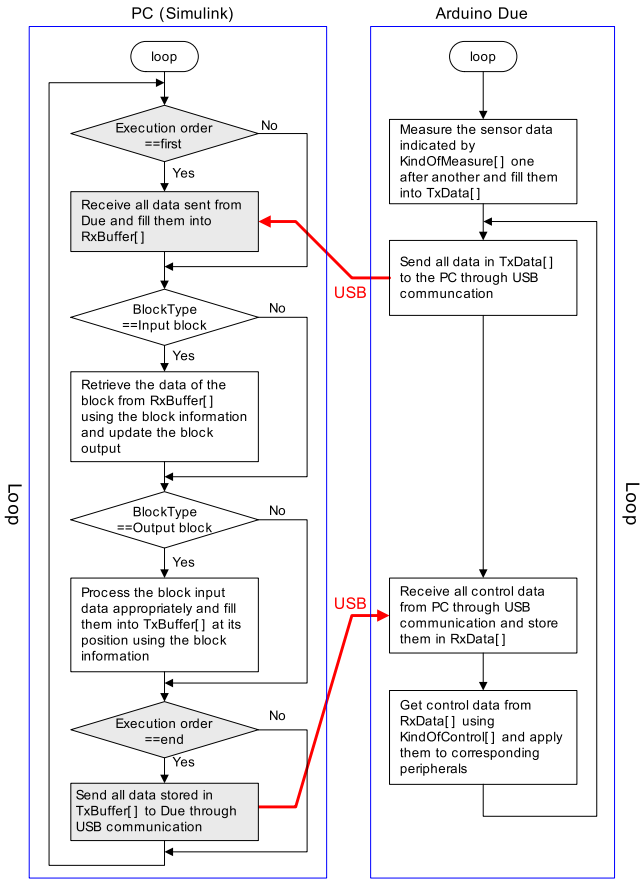

FIGURE 10. Flowcharts for the loop phase.

Real-time control requires that all control operations are finished within the given sample time. In the proposed RCP system, the control computation is performed in Simulink running under Windows, which is not a real-time operating system. Furthermore, the USB communication is not real-time either. Hence, the operations S1 to S5 in one iteration of the control loop are not guaranteed to be finished within a given sample time. In this case, the actual sample time should be carefully taken into account in the control algorithm. The Time block supported by the proposed RCP system enables the control designer to construct a control algorithm that can handle a variable sample time.

The PC-side program can be implemented on CACSD tools other than MATLAB/Simulink as long as they support a mechanism for custom blocks. The program for the Arduino Due can be implemented on other open source hardware such as Beaglebone Black [12] or Nucleo board [25] with a high-speed USB interface.

IV. CONTROL IMPLEMENTATION AND SUPPLEMENTAL EXPLANATION

In order to illustrate that a control system can easily be implemented using the proposed RCP system, we construct a wireless control system for a two-wheeled mobile robot. Figure 11 is a schematic diagram of the control system of a two-wheeled mobile robot. Figure 12 shows a lab-built

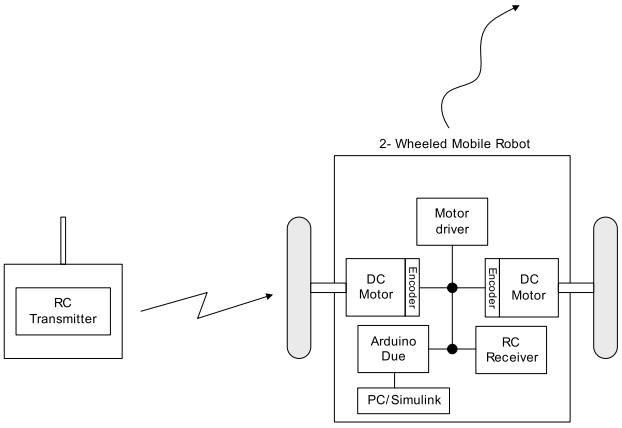

FIGURE 11. Schematic diagram of a two-wheeled mobile robot control using the proposed RCP system.

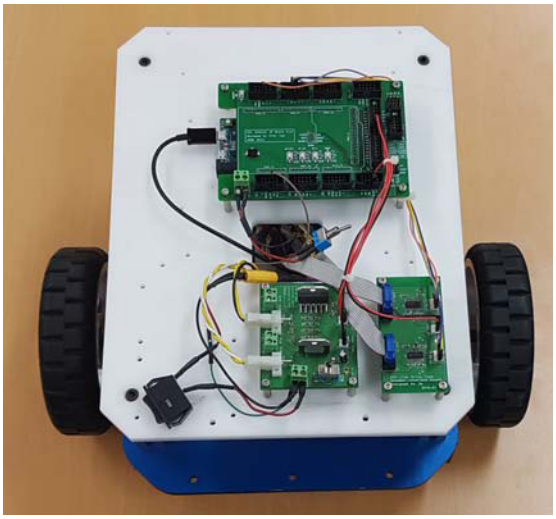

FIGURE 12. A lab-built mobile robot with the proposed RCP system mounted on it.

mobile robot used for experiments. The control purpose is to maneuver the robot by wirelessly changing the forward and rotational speeds of the robot. A robot operator transmits commands for the forward velocity and the rotational velocity of the robot using an RC transmitter. The RC receiver mounted on the robot receives these commands and outputs them as PWM signals. In the control system, the two velocity commands are calculated by measuring the widths of the PWM signals. Then, the two commands are converted into the velocity commands of the two wheels using the kinematic information for the robot. The forward velocity v and the rotational velocity w of a two-wheeled mobile robot are related with the wheel velocities w_R (right wheel) and w_L (left wheel) as follows [26]:

$$v = \frac{r(w_R + w_L)}{2}, \quad w = \frac{r(w_R - w_L)}{2b}, \tag{1}$$

where r is the radius of a wheel and b is half the distance between the two wheels. Using the above relations, velocity commands for the wheels are obtained as follows:

$$\bar{w}_R = \frac{\bar{v} + b\bar{w}}{r}, \quad \bar{w}_L = \frac{\bar{v} - b\bar{w}}{r}, \tag{2}$$

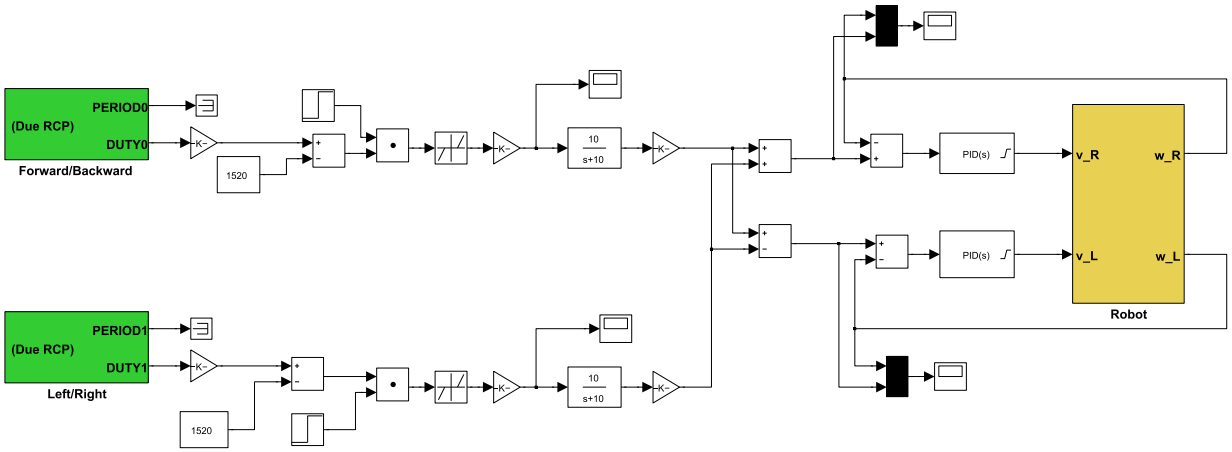

FIGURE 13. Control block diagram for a two-wheeled mobile robot.

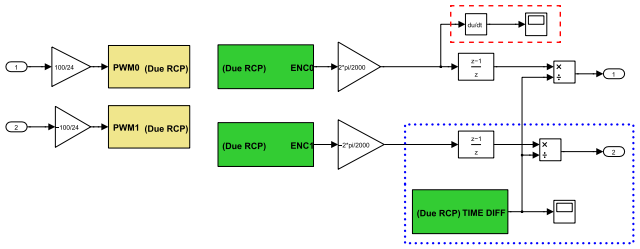

FIGURE 14. Block diagram of a subsystem for a two-wheeled mobile robot.

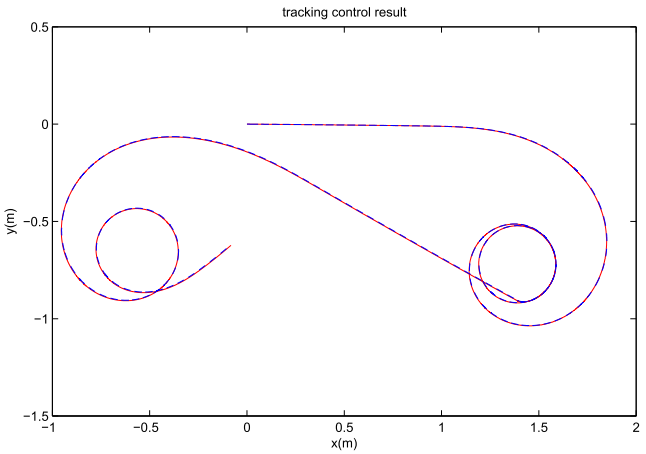

FIGURE 15. Tracking control result: reference(dashed) and actual position (solid).

where overbars denote commands. By including two PI velocity controllers in the control system, the operator can maneuver the robot according to the commands transmitted by the user. In the proposed RCP system, a PC performs control computations through Simulink. Thus we use Intel's NUC computer, which is a small-sized mini-PC, so that it can be mounted on a mobile robot.

Fig. 13 shows a Simulink block diagram for the control system constructed using the proposed RCP system. It is seen that the Period and Duty blocks are used to measure the widths of the PWM signals. The block labeled 'Robot' is a subsystem that drives two DC motors and measures the velocities of the motors. A more detailed internal configuration is shown in Fig. 14. We can see that two PWM blocks, two Encoder blocks, and a Time block are used. As shown in Fig. 13, it is seen that the control algorithm can be easily and quickly implemented by using functional blocks provided by Simulink, such as a transfer function block or a PID block together with the I/O blocks provided in the proposed RCP system. Any signal can also be monitored using the Scope block. This ease of monitoring provides great convenience in controller design. The implemented controller in Fig. 13 performed satisfactorily. As shown in this example, this easy-to-use controller design process allows students to concentrate on the theoretical aspects of controller design and avoid the hassles of manual coding, which are prone to errors.

Fig. 15 shows the experimental results when tracking control experiments are performed on the mobile robot using the control algorithm shown in Fig. 13. The sample rate of the control algorithm is 1 KHz. The blue dashed line shows the reference position and the solid red line indicates the actual position. We see that the mobile robot successfully follows the reference position.

In the above experiment, we used two PI controllers for simple illustration. However, more advanced control methods have been widely used for robotic systems. For example, adaptive control/neural network control for robotic systems has gained much attention recently [27]–[30]. The control methods presented therein can also be implemented with the proposed RCP system.

In the remaining part of this section, we add some supplemental explanations for Section III using the Simulink model in Fig. 13. When we run the Simulink model in Fig. 13, the information on the input/ouput blocks is collected through the initialization phase shown by the flow chart in Fig. 9. The information collected includes the order in which each

TABLE 2. Execution order determined after the initialization phase.

Excution order	Block type and channel
0	Period,duty channel 0
1	Period,duty channel 1
2	Time difference
3	Encoder counter, channel 0
4	Encoder counter, channel 1
5	PWM channel 0
6	PWM channel 1

TABLE 3. TxDataInfo and RxDataInfo obtained after the initialization phase.

index	RxDataInfo
0	Period,duty channel 0
1	Period,duty channel 1
2	Time difference
3	Encoder counter, channel 0
4	Encoder counter, channel 1

index	TxDataInfo
0	PWM channel 0
1	PWM channel 1

TABLE 4. Offset information obtained after the initialization phase.

Input block type and channel	Offset
Period,duty channel 0	0
Period,duty channel 1	8
Time difference	16
Encoder counter, channel 0	20
Encoder counter, channel 1	24

Output block type and channel	Offset
PWM channel 0	0
PWM channel 1	4

I/O block is executed, with RxDataInfo indicating the type of input block and the order of execution among the input blocks and TxDataInfo indicating the type of output block and the order of execution among the output blocks. Also, in order to assemble the data to be batch-transferred, the offset information indicating where each block data resides in the buffer array is also obtained during the initialization phase. The information gathered is sent to Arduino Due so that Arduino Due can also perform its required initialization phase. TABLE 2 shows the execution order of the I/O blocks included in the Simulink model in Fig. 13. Among the multiple I/O blocks used, it is observed that the Period and Duty block with channel 0 has the first execution order. TABLE 3 shows the contents of TxDataInfo and RxDataInfo and Table 4 shows the offset information. According to Table 4, we find that data of Encoder block with channel 0 is stored at the index 20 in the buffer array. Using the offset information, the data can be assembled before the batch transfer at each loop. Data arrangement in the buffer array is shown in Fig. 16. These data are batch-transferred each time the loop is executed. By adopting the batch transfer strategy, the proposed RCP system can minimize the latency caused by the data transfer and consequently increase the sample rate of the control system.

The control system built from the proposed RCP system does not guarantee perfect hard real-time feature because it

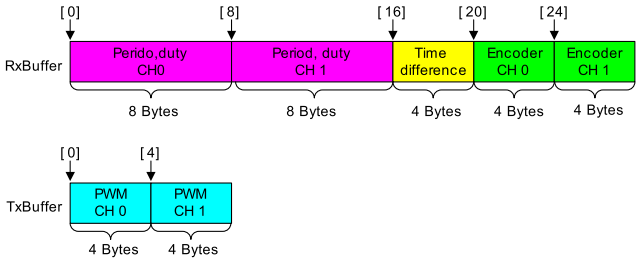

FIGURE 16. Buffer arrays (TxBuffer and RxBuffer) for batch transfer.

adopts MS Windows and USB communication, which are easily available but not real-time. Therefore, it may happen that the control loop is not finished within the assumed sample time. To solve this problem due to a variable sample time, the controller designer can use the Time block to obtain the actual sample time information. To have a better idea of this, refer to the Simulink model in Fig. 13. The control system in Fig. 13 includes two PI velocity controllers. The block diagram for calculating the velocity can be found in Fig. 14. It has two implementations for determining the angular velocity of the motor: the one in a dashed box and the other in the dotted box. The derivative block in a dashed box calculates the angular velocity, assuming the sample time to be constant. On the other hand, the blocks contained in dotted boxes obtain the angular velocity using the actual sample time length from the Time block. If the control system is hard real-time, the actual sample time length is constant and the two implementations will result in the same velocity values. However, this is not the case because of MS Windows and the USB communication.

Let us now consider a case where a sinusoidal velocity reference is applied to the PI velocity controller. We choose a sample rate of 4 KHz, which is 4 times faster than the previous example. Fig. 17 shows the length of the actual sample time measured by the Time block. It is seen that the sample time fluctuates around 0.25 ms. Deviation away from 0.25 ms is as high as 0.45 ms at 3.5 second, which shows that the control system is not hard real time. Velocity values obtained through the derivative block, which assume that the sample time length is constant, will be different from the actual velocity. Consequently, the velocity control performance is greatly deteriorated, as shown in Fig. 18. In the figure, the black line represents the reference velocity and the red line represents the velocity. The discrete values of the velocity are due to the quantization that occurs because the resolution of the encoder is finite. On the other hand, if the velocity is calculated using the Time block, the velocity is calculated by taking the actual sample time length into account. Even though the sample time length is variable, velocity information with good accuracy can be obtained. As a result, good control performance is also achievable. Fig. 19 shows the results of the velocity tracking control obtained with the Time block used for velocity calculation. It shows that tracking control is performed adequately. Unlike Fig. 18, it is observed

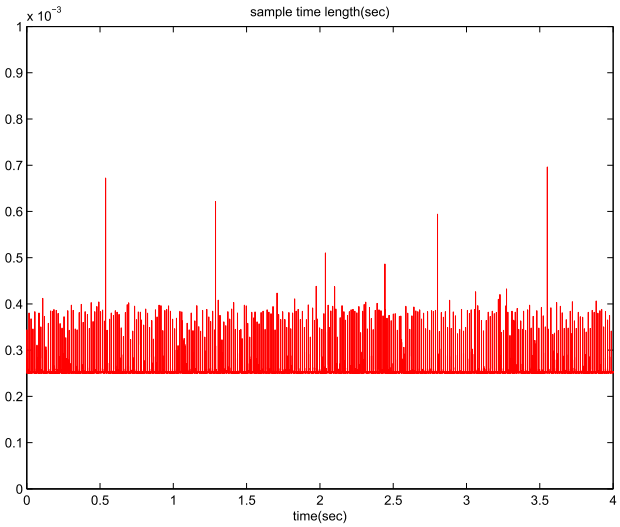

FIGURE 17. Measured sample time length at the sample rate of 4 KHz.

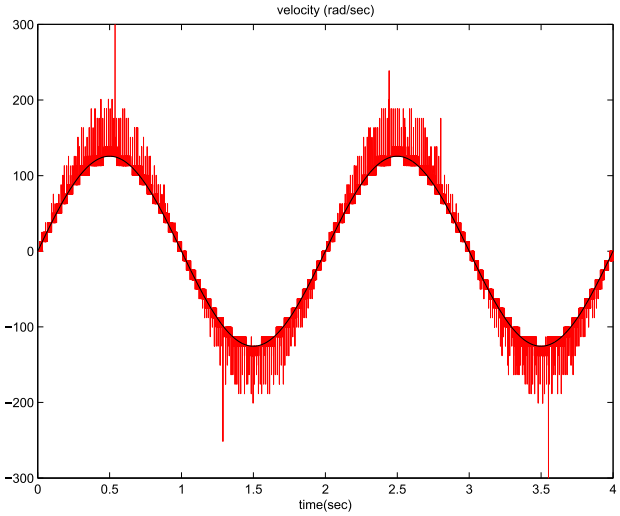

FIGURE 18. Velocity tracking performance with the derivative block used: reference (black line), velocity (red line).

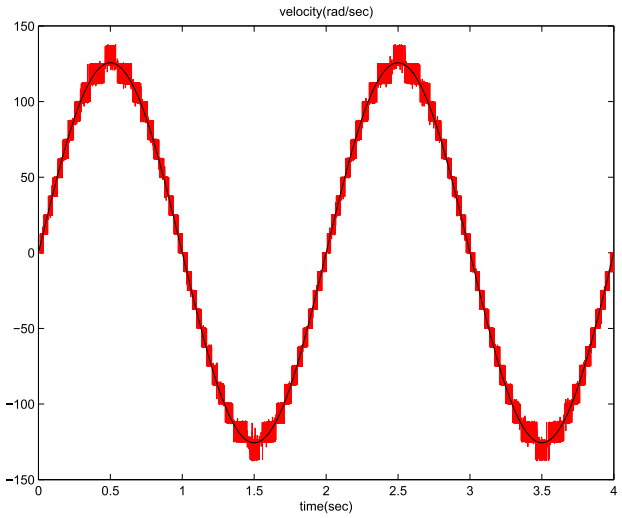

FIGURE 19. Velocity tracking performance with the Time block used: reference (black line), velocity (red line).

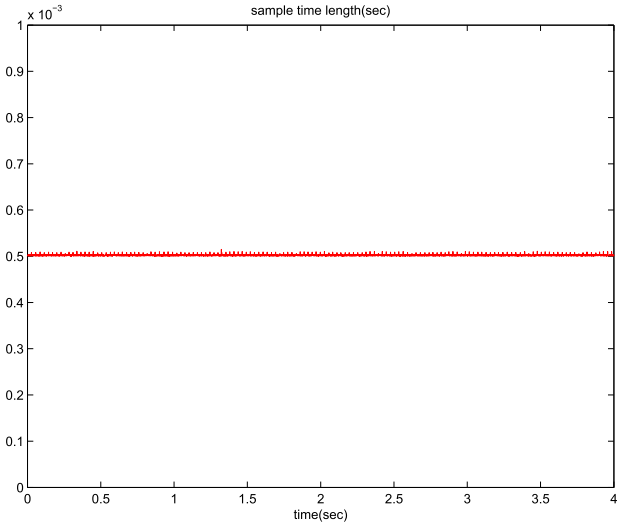

FIGURE 20. Measured sample time length at the sample rate of 2 KHz.

that the velocity tracks the reference input quite well. As seen from this example, the Time block provided in the proposed RCP system enables the control algorithm to effectively cope with a variable sample time. Because of this, it is also possible to choose the sample time length not too conservatively.

Another experiment was performed with a sample rate of 2 KHz to verify whether the proposed RCP system is real-time for that sample rate. Fig. 20 is the actual sample time length measured through the Time block. It is seen that the length is almost constant at 0.5 ms. This implies that the proposed RCP system can successfully be used in a control system with sample rates up to 2 KHz almost in real time.

V. HIGHLIGHTS AND COMPARISON

In this section, we highlight various feature of the proposed RCP system and compare them with existing RCP tools. For that purpose, the following criteria are taken into account.

- generation of C code
- required software and hardware
- price
- achievable sample rate of control loop
- mobility
- portability
- appropriateness for mass education

Having mobility means that the RCP-based controller can be mounted on a mobile platform and having portability means that the RCP system, including its hardware, can be easily carried, even to one's homes for take-home experiments or design. We assume that MATLAB/Simulink and a PC are common minimum requirements for all RCP systems considered below. Therefore, the price of the system are considered only for additional software and hardware.

TABLE 5. Comparison with existing RCP systems : ○ (good), △ (medium), × (bad).

	Target HW	I/O unit	SW	Max sample rate	Portability	Mobility	Price
Proposed RCP	PC	Arduino Due (USB interface)	I/O block library in Fig 4. Firmware for Arduino Due	2 KHz	○	△	Low
Simulink Desktop Real-Time	PC	I/O card (PCI, ISA, Express PCI, PCMCIA)	Windows Real-Time Simulink Coder	1KHz/20 KHz	X	X	High
xPC Target	PC	I/O card (PCI, ISA, Express PCI)	xPC Target Simulink Coder	50 KHz	X	X ○ (PC104)	High
dSPACE System	Single board or modular board (Ex: DS1004, MicroAutoBox)		Simulink Coder RT Interface ControlDesk	50 KHz	X	△	Very High
Embedded Targets	Embedded microcontroller board		Embedded Coder Target Support Package Compiler	Depends on processors	△	○	High

A. HIGHLIGHTS OF THE PROPOSED RCP SYSTEM

The required software and hardware for the proposed RCP system are the I/O block library shown in Fig. 4 and an Arduino Due board with the proposed program on the flash memory. The proposed RCP system does not require an expensive C code generator, known as a Simulink Coder. The price of a Simulink Coder is several thousand dollars. Since a Simulink model running on a PC in normal mode performs as a real-time controller, all features of MATLAB/Simulink are supported. A real-time kernel is not required because it utilizes the SysTick timer in the Arduino Due for real-time operation. However, the achievable sample rate is rather slow (about 2 KHz) as illustrated in the previous section. A low achievable sample rate can limit its applicability. However, it is useful for educational applications. Since a PC is used as a target hardware, even the control algorithm which is computation-intensive or requires a large amoung of memory can be well handled. For example, predictive control in [31], which solves the computation-intensive quadratic programming problem at each sample time, can be well handled. Swing-up control for a double inverted pendulum presented in [32], which requires a large amount of memory for storing all of the gain matrices in the horizon, can also be well handled. Since the proposed RCP system utilizes palm-sized Arduino Due as an I/O board, the portability is very good. The price of an Arduino Due is less than 50 dollars. With low cost and good portability, the proposed RCP system is perfect for mass education. Take-home design assignments can be well supported by the proposed RCP system. Nowadays, since various mini-PCs with very small sizes, such as the Intel NUC and HP Pavilion, are available, the proposed RCP system can be mounted on a mobile platform, having a medium level of mobility.

B. COMPARISON

In this subsection, a comparison with existing RCP systems is given. Readers may refer to [5] for a good survey on existing RCP tools. TABLE 5 compares the proposed RCP system with existing ones.

1) SIMULINK DESKTOP REAL-TIME

Simulink Desktop Real-Time provides a real-time kernel for executing Simulink models on a laptop or desktop running Windows or Mac OS X. It can be used in two ways. First, it can be used such that the Simulink model itself performs real-time control just like in the proposed RCP system. As a result, no code generation is required. In this case, Simulink Desktop Real-Time supports real-time performance up to 1 kHz, which is below that of the proposed RCP system. Second, it can be used such that C code from the Simulink model is generated for the same computer using Simulik Coder. The compiled executable runs in Windows Kernel mode. In this case, Simulink Desktop Real-Time supports real-time performance up to 20 KHz. Simulink Desktop Real-Time also includes library blocks that connect to a wide range of I/O cards with various interfaces such as PCI, ISA, PCI Express, and PCMCIA. Unfortunately, no I/O device supports the USB interface. As a result, the portability is very poor. Since mini-PCs do not support slot-based interfaces, Simulink Desktop Real-Time doesn't have good mobility. Furthermore, the price of I/O devices ranges from several hundred to several thousand dollars. For these reasons, Simulink Desktop Real-Time is not appropriate for mass education.

2) xPC TARGET

In xPC Target, two separate computers are required. A host computer with Simulink generates C code for a PC compatible target hardware with its own real-time operating system. Therefore, Simulink Coder is required. The host connects to the target using TCP/IP during operation for visualization of the results. Since the computer is completely dedicated to xPC Target tasking, xPC Target can achieve sample rates approaching 50 KHz, depending on the processor performance level, model size, and I/O complexity. xPC Target

supports a similar set of I/O cards compared to Simulink Desktop Real-Time, which means that no I/O cards with USB interfaces are available and thus portability is poor. However, xPC Target allows mobile operation by using a target computer in a PC/104 form factor. PC/104 I/O modules are also available. The price of PC/104 modules ranges from several hundred to more than 1000 dollars, showing that xPC Target is not appropriate for mass education.

3) dSPACE SYSTEM

The dSPACE system is an RCP system provided by Company dSPACE Gmbh. In the dSPACE system, the Simulink Coder together with Real Time Interface (RTI) generates code for high-performance hardware based on PowerPC processors. A single board (DS1104) or modular hardware (processor board + I/O boards) runs the compiled application in real-time. The ControlDesk software on a host computer interacts with the application running on a target to visualize results. The achievable sample rate is 50 KHz, depending on the processor performance level, model size, and I/O complexity. The price of the system is very high, including both the software and the hardware. If MicroAutoBox hardware is used as a target, it is mobile even though the size is not small enough. Due to its high price and performance, the dSPACE system is appropriate for advanced research.

4) EMBEDDED TARGETS

An embedded microcontroller board can be used as a target hardware. In Embedded Targets, C code is generated from the Simulink model for a particular embedded microcontroller using the Embedded Coder, with prices as high as several thousands of dollars. The Target Support Package (TSP) supports the complete code generation process for a particular microcontroller. The Integrated development environment (IDE) for the microcontroller is required for compilation of the generated code. The JTAG debugger is needed for downloading the compiled application to the embedded microcontroller board. Embedded microcontrollers have limited computational power and memory compared to PCs. Therefore, the achievable sample rate depends on the microcontroller performance, model size, and I/O complexity. Computation-intensive control algorithms will drop the achievable sample rate drastically. Even though the embedded mircocontroller board is portable, take-home assignments can be hardly supported because of the high price of Embedded Coder. Since it is based on a embedded target, it has the best mobility among all RCP systems.

VI. CONCLUSIONS

In this paper, we proposed a new RCP system based on MATLAB/Simulink and the Arduino Due with a built-in high-speed USB communication interface.

The proposed RCP system currently provides ten I/O library blocks for use with Simulink. Using those library blocks, users can build a prototype control algorithm both easily and quickly. We described the programs for both sides, which are the Arduino side and PC side. Since the control algorithm is computed by Simulink running on a PC, we can make full use of the strength of Simulink. Furthermore, one can use various functions provided by MATLAB, and thus complicated control algorithms can be built. Since the proposed RCP system does not use a C code generator, the price of which is several thousands of dollars, and since it uses the cheap Arduino Due board as an I/O device, it is ideal for mass education for engineering undergraduates. Furthermore, the small size of an Arduino Due board enables take-home assignments to be supported. The proposed RCP system has limited real-time property and can be used at a moderate sample rate of 2 KHz. Despite this limitation, several advantages such as cost effectiveness, portability, and good availability of the hardware make it sufficiently useful for educational purposes. In the future, we plan to add more I/O blocks so that the proposed RCP system can handle more control systems. We will also consider using the USB 3.0 SuperSpeed interface, which is ten times faster than the USB 2.0 high-speed interface, in order to increase the sample rate of the control loop.

REFERENCES

[1] M. Deppe, M. Zanella, M. Robrecht, and W. Hardt, "Rapid prototyping of real-time control laws for complex mechatronic systems : A case study," *J. Syst. Softw.*, vol. 70, no. 3, pp. 263–274, 2004.

[2] R. Isermann, "Mechatronic systems—Innovative products with embedded control," *Modern Instrum.*, vol. 16, no. 1, pp. 14–29, 2008.

[3] W. Lee, M. Shin, and M. Sunwoo, "Target-identical rapid control prototyping platform for model-based engine control," *Proc. Inst. Mech. Eng. D, J. Automobile Eng.*, vol. 218, no. 7, pp. 755–765, 2004.

[4] D. Hercog and K. Jezernik, "Rapid control prototyping using matlab/simulink and a DSP-based motor controller," *Int. J. Eng. Ed.*, vol. 21, no. 3, pp. 1–9, 2005.

[5] R. Grepl, "Real-time control prototyping in MATLAB/Simulink : Review of tools for research and education in mechatronics," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Des.*, Dec. 2011, pp. 881–886.

[6] S. Rebeschiess, "MIRCOS-Microcontroller-based real time control system toolbox for use with matlab/simulink," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Des.*, Dec. 1999, pp. 267–272.

[7] R. Bucher and S. Balemi, "Rapid controller prototyping with matlab/simulink and Linux," *Control Eng. Pract.*, vol. 14, no. 2, pp. 185–192, 2006.

[8] S. Grzegorz, Z. Tomasz, and B. Andrzej, "Rapid control prototyping with Scilab/Scicos/RTAI for PC-based ARM-based platforms," in *Proc. Comput. Sci. Inf. Technol. (IMCSIT)*, Oct. 2008, pp. 739–744.

[9] Y. S. Lee, J. H. Yang, S. Y. Kim, W. S. Kim, and O. K. Kwon, "Development of a Rapid Control Prototyping System Based on Matlab and USB DAQ boards," *J. Inst. Control, Robot., Syst.*, vol. 18, no. 10, pp. 912–920, 2012.

[10] M. Margolis, *Arduino Cookbook*, 2nd ed. Sebastopol, CA, USA: O'Reilly, 2012.

[11] R. Toulson and T. Wilmshurst, *Fast and Effective Embedded Systems Design*. Waltham, MA, USA: Elsevier, 2012.

[12] D. Molly, *Exploring Beaglebone*. Hoboken, NJ, USA: Wiley, 2015.

[13] *The Official Website of Open Source Hardware Association*, accessed on Apr. 15, 2017. [Online]. Available: https://www.oshwa.org/definition/

[14] G. Senthilkumar, K. Gopalakrishnan, and V. S. Kumar, "Embedded image capturing system using Raspberry Pi system," *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 3, no. 2, pp. 213–215, 2014.

[15] M. Z. Hoz, L. Acho, and Y. Vidal, "An experimental realization of a chaos-based secure communication using Arduino microcontrollers," *Sci. World J.*, vol. 2015, 2015, Art. no. 123080.

[16] A. K. Dennis, *Raspberry Pi Home Automation With Arduino*. Birmingham, U.K.: Packt Publishing, 2013.

[17] T. Baden, A. M. Chagas, G. Gage, T. Marzullo, L. Prieto-Godino, and T. Euler, "Open labware: 3-D printing your own lab equipment," *PLoS Biol.*, vol. 13, no. 3, p. e1002086, 2015.
[18] D. K. Fisher and P. J. Gould, "Open-source hardware is a low-cost alternative for scientific instrumentation and research," *Modern Instrum.*, vol. 1, no. 2, pp. 8–20, 2012.
[19] *Matlab Official Home Page*, accessed on Apr. 15, 2017. [Online]. Available: http://www.mathworks.com
[20] *Labview Official Home Page*, accessed on Apr. 15, 2017. [Online]. Available: http://www.ni.com/labview
[21] *Scicos Official Home Page*, accessed on Apr. 15, 2017. [Online]. Available: http://www.scicos.org/
[22] *Cemtool Official Home Page*, accessed on Apr. 15, 2017. [Online]. Available: http://www.cemtool.co.kr/
[23] *Developing S-Functions*, MathWorks Inc., Natick, MA, USA, 2017.
[24] *The Website of Atmel Studio*, accessed on Apr. 15, 2017. [Online]. Available: http://www.atmel.com/tools/atmelstudio.aspx
[25] C. Noviello, *Mastering STM32*. Victoria, BC, Canada: Leanpub, 2017.
[26] S. G. Tzafestas, *Introduction to Mobile Robot Control*. Waltham, MA, USA: Elsevier, 2014.
[27] W. He, Y. Chen, and Z. Yin, "Adaptive neural network control of an uncertain robot with full-state constraints," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 620–629, Mar. 2016.
[28] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 46, no. 3, pp. 334–344, Mar. 2016.
[29] W. He, Y. Ouyang, and J. Hong, "Vibration control of a flexible robotic manipulator in the presence of input deadzone," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 48–59, Feb. 2017.
[30] W. He and Y. Dong, "Adaptive fuzzy neural network control for a constrained robot using impedance learning," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/7865986/, doi: 10.1109/TNNLS.2017.2665581.
[31] Y. S. Lee, G. Y. Gyeong, and J. H. Park, "QP solution for the implementation of the predictive control on microcontroller systems and its application method," *J. Inst. Control, Robot. Syst.*, vol. 20, no. 9, pp. 908–913, 2014.
[32] K. Graichen, M. Treuer, and M. Zeitz, "Swing up of the double pendulum on a cart by feedforward and feedback control with experimental validation," *Automatica*, vol. 43, no. 1, pp. 63–71, 2014.

YOUNG SAM LEE received the B.S. and M.S. degrees in electrical engineering from Inha University, Incheon, South Korea, in 1999, and the Ph.D. degree in electrical engineering from Seoul National University, South Korea, in 2003. From 2003 to 2004, he was a Senior Researcher with Samsung Electronics Co. Since 2004, he has been with the Department of Electrical Engineering, Inha University. He is the author of four books and more than 50 articles. His research interests include computer-aided control system designs, rapid control prototyping, control and instrumentation, robot engineering, and embedded systems.

BONGEON JO received the B.S. and M.S. degrees in electrical engineering from Inha University, Incheon, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical engineering. From 2014 to 2016, he was a Laboratory Assistant with the Department of Electrical Engineering, Inha University. His research interest includes rapid control prototyping, control of powered lower limb prosthesis, and embedded systems.

SOOHEE HAN (M'12–SM'13) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University (SNU), Seoul, South Korea, in 1998, 2000, and 2003, respectively. From 2003 to 2007, he was a Researcher with the Engineering Research Center for Advanced Control and Instrumentation, SNU. In 2008, he was a Senior Researcher with the Robot S/W Research Center, Seoul. From 2009 to 2014, he was with the Department of Electrical Engineering, Konkuk University, Seoul. Since 2014, he has been with the Department of Creative IT Engineering, Pohang University of Science and Technology, Pohang, South Korea. His research interests include computer-aided control system designs, distributed control systems, time-delay systems, and stochastic signal processing.