A Comparison of Platform Configurations for Robotics Development within ROS2, Raspberry Pi, and Webots

Ms. Katherine Gisi, Iowa State University
Dr. Diane T. Rover, Iowa State University

Diane Rover is a University Professor of Electrical and Computer Engineering at Iowa State University. She has held various faculty and administrative appointments at ISU and Michigan State University since 1991. She received the B.S. in computer science in 1984, and the M.S. and Ph.D. in computer engineering in 1986 and 1989 (ISU). Her teaching and research has focused on embedded computer systems, reconfigurable hardware, parallel and distributed systems, visualization, performance monitoring and evaluation, and engineering education. She has held officer positions in the ASEE ECE Division, served as an associate editor for the ASEE Journal of Engineering Education, and served on the IEEE Committee on Engineering Accreditation Activities, the IEEE Education Society Board of Governors, the ABET EAC (2009-2014), and EAC Executive Committee (2015-2018). Dr. Rover is a Fellow of the IEEE and of ASEE.

Dr. Phillip H Jones III, Iowa State University of Science and Technology

Phillip H. Jones received his B.S. degree in 1999 and M.S. degree in 2002 in electrical engineering from the University of Illinois, Urbana-Champaign. He received his Ph.D. degree in 2008 in computer engineering from Washington University in St. Louis. Currently, he is an Associate Professor in the Department of Electrical and Computer Engineering at Iowa State University, Ames, where he has been since 2008. His research interests are in adaptive computing systems, reconfigurable hardware, embedded systems, and hardware architectures for application specific acceleration. Jones received Intel Corporation sponsored Graduate Engineering Minority (GEM) Fellowships from 1999-2000 and from 2003-2004. He received the best paper award from the IEEE International Conference on VLSI Design in 2007.

A Comparison of Platform Configurations for Robotics Development using ROS2, Raspberry Pi, and Webots

Abstract

Background: The field of robotics continues to expand with new, progressive technologies. The vast ecosystem of robotics, with its countless routes, is initially challenging to navigate. It is even more so during a season of virtual instruction. Creating strategic entrance points for the incoming students merits consideration.

Purpose: Our aim is to provide a way for students to explore three critical areas in robotics: the physical robot (hardware), the simulated robot (digital twin), and the operating system and code (controller).

Method: We explore options for integrating two advanced robot development frameworks (i.e. Robot Operating System 2 (ROS2) and Webots) in combination with the widely used beginners computing platform (i.e. Raspberry Pi) to facilitate introducing students to the field of robotics, while also providing them a path to advanced robot topics.

Results: Two configurations for integrating ROS2, Webots, and the Raspberry Pi are presented. The advantages and concerns for each are discussed, along with pointers for mitigating concerns.

Conclusion: The integration of these frameworks and platform, along with virtualizing the physical robot as a digital twin, opens a gateway for students to enter the field of robotics. We also envision our ideas supporting paths for improving online distance learning, developing robot networks, and reducing field testing time.

Keywords: Digital twin, embedded systems, online robotics education, simulation, virtual instruction

1. Introduction

1.1 Incentive

Robotics has proven to be an attractive and powerful way to teach the fundamentals of engineering, hardware systems, and coding (Liang-Yi Li, 2009). It is hands-on and provides instant feedback, encouraging exploration through trial and error. Perhaps more importantly, in the marketplace, embedded systems and Internet of Things (IoT) technologies are flourishing, indicating huge amounts of data will be continuously gathered. Subsequently, there must be applications for this data. In the future, hopefully, this information will be processed and used by an automated service provider which can take many forms, robotics being a primary option (Batth, Nayyar, & Nagpal, 2018). The global robotics industry is growing at a compound annual growth rate in the teens or greater. (Tobe, 2017).

1.2 Purpose

With robotics and embedded systems holding promise for the fundamentals and the future, it is advantageous to find new and innovative methods to teach and to foster interest in the field (Gennert & Putnam, 2018). In pursuit of developing a foundational appreciation of and competency using embedded systems via robotics for engineering students, a practical arrangement of three modern robotics platforms is being investigated with an emphasis on distance learning. With the challenges of virtual classes being universally felt, each platform must demonstrate capacity for distance learning and continued exploration. Taking steps to utilize simulation when face-to-face instruction is not viable is a driving objective in this research.

1.3 Educational Context

The educational context for this study of robotics platforms is the Introduction to Embedded Systems course in the Department of Electrical and Computer Engineering at Iowa State University. The 200-level course is required for computer, cybersecurity, and electrical engineering majors, and for software engineering, it is one of two courses students choose from to fulfill a requirement. Students learn about embedded systems concepts and design using a robotics platform in the laboratory. The mobile robot in the lab is built around the iRobot Create 2 (Roomba).

The course introduces students to hardware and software aspects of embedded systems including microcontrollers, memory-mapped input/output, input/output interfaces, embedded programming in C, initialization and configuration of peripherals in software, general purpose input/output (GPIO) ports, polling and interrupt processing, serial communication (UART), analog-to-digital conversion (ADC), hardware timers (GPTM), input capture, pulse-width modulation, sensors, servo motors, mobile robots, and object detection. The first third of the course covers foundational concepts and skills; the middle third, understanding and using microcontroller peripherals (GPIO, UART, ADC, GPTM modules); and the final third, implementing a project in the lab for an autonomous vehicle application. The course has three lecture hours and two lab hours each week. The weekly labs are guided by undergraduate and graduate teaching assistants. The lecture and lab content and flow are highly integrated. The final project is introduced early in the semester and phased in through class and lab activities prior to exclusively working on it in lab.

The mobile robot in the lab can be controlled with Open Interface commands from a Texas Instruments LaunchPad microcontroller board to the Roomba robot. The microcontroller board interfaces to input/output devices added to the robot, including an infrared sensor, ultrasonic sensor, and servo motor (used to get scans of sensor readings). The entire robotics platform is referred to as the CyBot (named after the Iowa State Cyclones) and is shown in Figure 1.

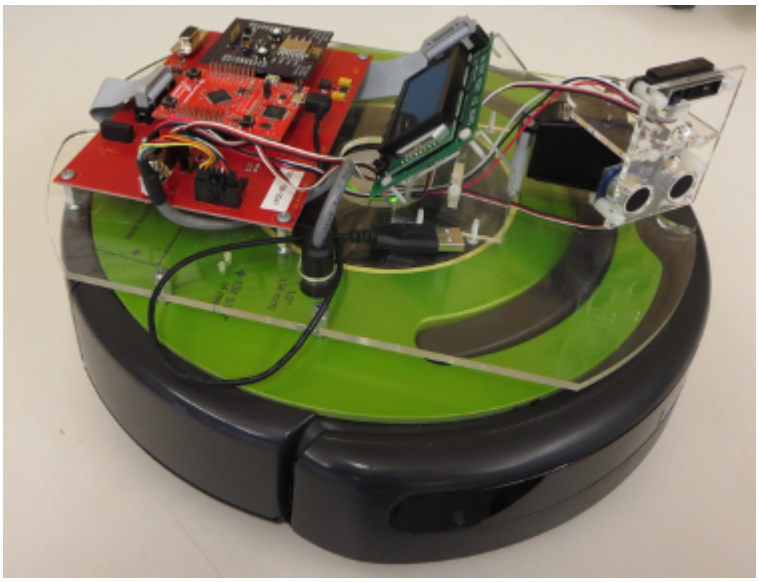

Figure 1: CyBot robotics platform in the Introduction to Embedded Systems course.

In the final project, teams program the microcontroller to move the robot through a test field and avoid obstacles to reach a destination. Teams have the option to select a design problem for a Mars rover autonomous vehicle application, in which the primary task of the rover is to navigate through hazardous terrain to a retrieval zone where it will send data to mission control. The design criteria for the Mars rover application are predefined as shown in Table I.

TABLE I. Design Criteria for the Mars Rover Application

Criterion	Description
Reach goal	Find and stop in the zone where it can transmit the data it has collected.
Object detection	Detect surface objects, such as boulders and stalagmites, to avoid collisions and damaging components.
Boundary detection	Identify the boundaries of the "safe zone" to stay within areas with safe levels of solar radiation.
Object/boundary avoidance	Avoid objects it has detected and identified to limit damage.
Information display	Display information in a form that is readable to humans who are controlling and monitoring its activities.
Vehicle communication	Receive commands from mission control.
Reach goal quickly	Navigate the course and reach the goal area within a time limit so that it does not miss the transmission window.
Autonomous movement	Make decisions about movement without human commands (in case commands cannot be received).
Object identification	Determine the type of object it has detected to help determine how it will avoid the object.

With recent revisions to the course, students have more options and responsibility in project selection and management. More information about the redesign of the course project is found in (Rover, Fila, Jones & Mina, 2020).

In the midst of spring semester 2020, the COVID-19 pandemic made it infeasible for student teams, teaching assistants, and professors to safely work together in an in-person lab environment. In response, a high-fidelity virtualized CyBot platform was developed later in the semester, spear-headed by a senior in our department as an extension to their senior design project (https://sdmay20-04.sd.ece.iastate.edu). The virtualized platform is shown in Figure 2. There are two TI LaunchPad boards, i.e., the red boards in the figure. The board on the right is

programmed by students and acts as the controller as in the physical CyBot. However, without the physical CyBot, the rest of the system is virtualized, and the board on the left mimics components such as the sensors.

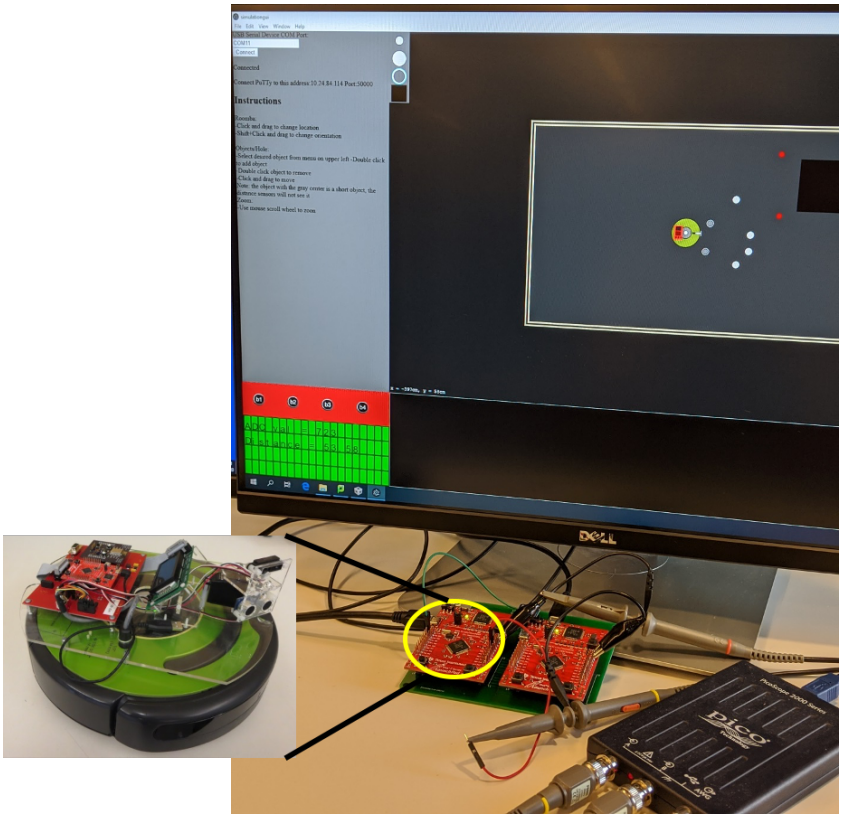

Figure 2: Virtualization of the CyBot platform.

The realism of the virtualized CyBot is such that embedded applications developed for the actual CyBot can be deployed to the virtual CyBot with no modification. The timing of this microcontroller-implemented virtualized Cybot is such that students can use an oscilloscope to probe sensor and actuator signals in real-time. The virtual CyBot platform was piloted with students in summer 2020, and continues to be used in fall 2020 to give students a fully remote infrastructure for developing and evaluating their embedded applications. Figure 3 shows a side by side comparison of the actual CyBot operating within a test field next to the virtual CyBot operation within a virtual test field.

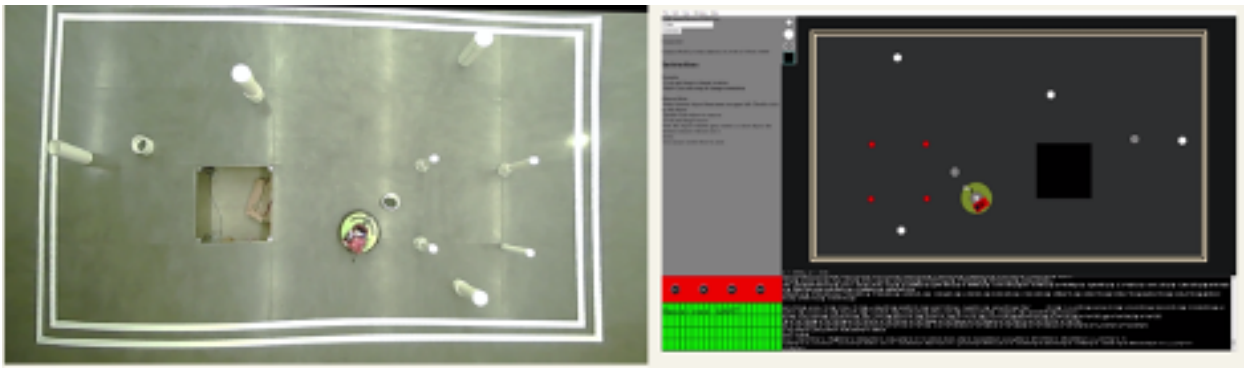

Figure 3: (Left) CyBot robot operating in a test field. (Right) Virtual CyBot in a virtual test field.

In effect, the need for a virtual lab platform due to COVID restrictions on in-person lab instruction has resulted in additional options to support hands-on learning of embedded systems and mobile robots. The virtual CyBot offers a digital twin of the physical CyBot. This opens new opportunities for learning. Consequently, we initiated the study presented in this paper to explore other robotics platform configurations with similar virtual capabilities.

2. Background and Related Work

2.1 State of Robotics Education and Robotics in Embedded Systems Courses

As mentioned in section 1.1, robotics is an effective way to motivate the next generation. Utilizing robots does not have to be confined to elementary classrooms or specialized robotics courses. In electrical and computer engineering embedded systems and microcontroller courses, using autonomous robots has been shown to improve students' comprehension of the essential concepts (Lessard, 1999). "When teaching advanced courses in microcontroller theory it is difficult to teach advanced concepts such as real time operating systems, multiple interrupts, fuzzy logic, and structured design techniques. Part of the difficulty is that a lot of the finer details of these concepts are not readily visible to the students. If paper exercises are used to illustrate these concepts, students often view them as esoteric, obtuse, and dry." (Beavis, et al., 2005) Texas A&M found success in strategies such as adopting a common microprocessor throughout a sequence of classes and providing students freedom in their robot's goals and design and is still investigating educational impacts and integrating industry trends. (Hur, Goulart, Porter, Sarker, & Willey, 2020).

2.2 Simulation and Distance Learning

The virtual classroom presents its own set of challenges. In robotics-based courses this is markedly felt, as their primary advantage lies in the hands-on and visual feedback components. Drushel & Gallagher have identified and listed a number of issues when teaching robotics courses online. These include communication challenges, restrictions on hardware design (they had previously been letting students design their own robots, but switched to a fixed hardware

for simulation purposes), and a necessary switch to a more difficult coding language. (Drushel & Gallagher, 2008). Robotics classes have been flipped with positive results. At Rose-Hulman their flipped class was evaluated similarly favorable to their traditional class (Berry C. A., 2017). However, they still had the hands-on robot component. Many of the communication challenges documented in online robotics courses are common within many virtual classrooms. Conversely, the issues specific to robotics in many virtual classrooms could stem from trying to force a face-to-face curriculum to an online platform. Perhaps a whole new approach is needed, starting with a focus on developing the simulation component.

2.3 Webots

Webots is a robotics simulation software with an intuitive and straightforward interface. The application, made open-source in 2018, offers the capability to contrive a customized robot simulation. Webots is interactive, allowing the user to apply forces while running a simulation, and has a wide variety of environments built-in, including: indoor, outdoor, water, and sky. Webots's primary advantages over other similar simulators is its physics engine, graphics rendering, and extensive libraries of sensors and actuators that make building and testing robots realistic and accessible (Micheal, 2004). Further, the simulation progresses via a virtual clock so time can be sped and slowed if desired. Webots is widely used in academia and industry from everything from biomedical micro-robots (Guo, Zhang, FAIMBE, & Man, 2012) to refining production line sorting (Pan, Ma, Mu, An, & Chen, 2018). Webots has a curriculum based on the e-puck robot. Analysis and feedback of this curriculum has show that Webots has potential to create an educational and explorative environment (Guyot & Rohrer, 2011)

2.4 The Robot Operating System

The Robot Operating System (ROS) is a light, open-source framework developed to standardize internal and external communication between robotic components (Quigley, et al., 2009). It consists of a group of libraries and packages for building reusable, language-independent robot applications. It utilizes peer-to-peer communication of specified nodes such as publisher, subscriber, service, and client nodes. It runs on top of a Linux Ubuntu operating system. Extensively used throughout education, research, and industry, ROS is moving toward becoming a standard in Robotics (Su, 2019). The ROS community and ecosystem is substantial, providing extensive coding and problem-solving support. A main asset of ROS is that it adds a single layer of abstraction, allowing a user without specialized technical skills and knowledge to approach advanced projects without limiting capability. It has been found to be an effective approach to teaching robotics concepts (Wilkerson P.E., Forsyth, Sperbeck, Jones, & Lynn, 2017).

ROS2, released in 2017, is an improved version of ROS, boasting greater robustness from changes in the fundamentals of the framework.

2.5 Raspberry Pi

Raspberry Pi, a minicomputer originally made for educational purposes, is now widely used and recognized. It is small, affordable, and adaptable making it a good entry-level option for IoT embedded systems (Meruje, 2018). Builds of the Raspberry Pi come with a range of ports and

connectivity including USB, HDMI, GPIO pins, Ethernet, CSI, and others. This gives the Pi user the ability to customize peripherals. Most notable are the GPIO pins, which allow a range of sensors and hardware to interface with the Pi.

2.6 Benefits of ROS2, Webots in confluence

Both ROS2 and Webots are powerful platforms that are used in cutting-edge areas of research and industry (Webots: Robot Simulator, n.d.). Their potential makes them an ideal software/framework for the future robotics engineer to work with. Starting in robotics with platforms geared toward beginners is a popular option and has its many advantages. However, the learner will eventually be constrained by software and/or hardware limitations such as expense, capability, or too many layers of abstraction. Moreover, many beginner robotics sets leave scarce room for creativity in that they are overly "boxed", that is, they provide step-by-step instructions and vision for the final product and offer few tools for creative deviation. ROS and Webots can work in tandem with any chosen hardware, if any hardware is desired at all, and do not limit creative capacity as they provide the building blocks rather than a start-to-finish "kit". For the entry-level learner, it could be initially challenging to learn these advanced applications. However, with the right configuration, a learner can advance without switching platforms. Additionally, the student will gain a deep knowledge of robotics simulation and ROS, which may open doors in research and industry. Investigating ROS and Webots with the intent to use them in an embedded systems classroom is a unique approach as far as our background research suggests.

3. Methods and Configurations

3.1 Preliminary Notes

When starting with ROS2, the Raspberry Pi is a beneficial selection for an onboard computer. In addition to the advantages listed in section 2.3, it has the capability to run the Linux kernel and the Ubuntu flavor of Linux which creates the option to run ROS2 directly on a robot versus running ROS2 externally. Each configuration has its advantages as explained in the next few sections.

3.2 ROS2 on Raspberry Pi

Running ROS2 directly on the Raspberry Pi, in addition to running ROS2 with Webots on an Ubuntu machine (desktop/laptop/virtual), is the more intuitive option within the structure of ROS. When ROS is operating on the Pi each attached peripheral can be treated as an individual node, creating a well-defined, distinct structure. Furthermore, the intelligence can be coded and executed directly on the Pi. This provides the option to run the robotic device either connected or independent of another Ubuntu-ROS machine.

The Webots simulation is built on and run from an Ubuntu-ROS desktop/laptop/virtual environment and controlled via the external controller option in the Webots application. By cloning the robot's ROS workspace from the Pi to the computer running the Webots simulation

and changing the robot's driver functions in the controller code to Webots specific functions, a cross-compilation type system could be developed. This process is made more feasible when referring to the cross-compilation examples and documentation provided by Webots (Transfer to Your Own Robot - Webots documentation, n.d.). The webots_ros2 package should be cloned from the Cyberbotics repository into the ROS2 workspace on the computer, and the files should be organized and written in accordingly. With this system in place, the simulated robot and the physical robot should behave most similarly (with respect to the platform) because of the way both ROS and Webots structure a robot. See Figure 4.

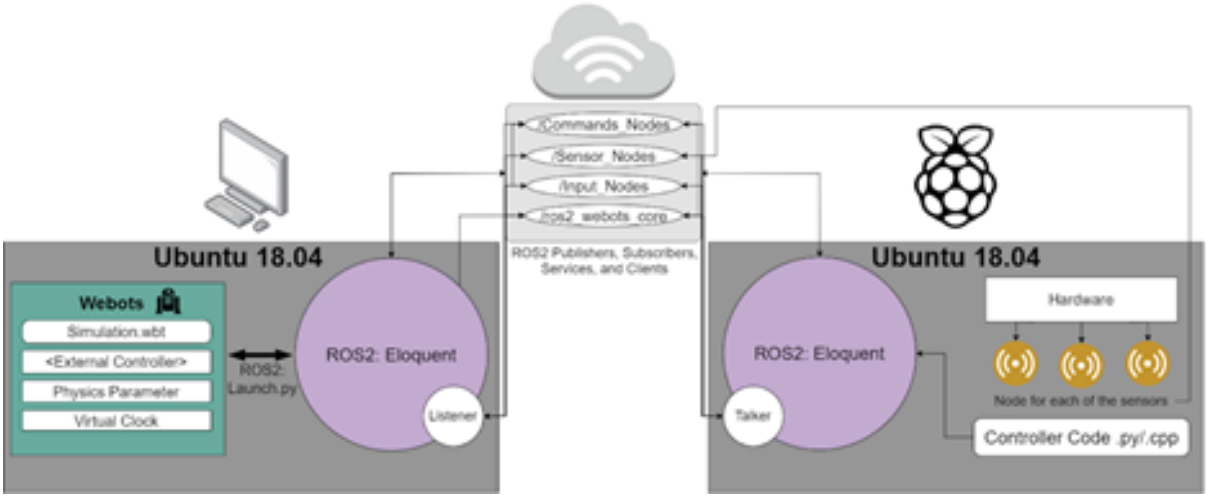

Figure 4: Configuration 1 details, the ROS network, and significant nodes.

3.3 ROS2 on Ubuntu Machine with Alternate Operating System on Raspberry Pi

Running ROS2 exclusively on an Ubuntu machine while installing an alternate operating system on the Raspberry Pi provides the option to make use of common Raspberry Pi libraries and the prewritten controller code that often accompanies beginner's robotics kits. This configuration eliminates the need for writing robot specific driver code, which offers a quicker route to experimentation with the physical robot. All communication can be sent or received within a single ROS message for each of the sensors or actuators respectively. The message containing the robot data is transmitted into the ROS workspace. Then the controller code, data collection, analysis, and similar services and clients can execute and publish to a single return message function. Although this approach is efficient, executing the intelligence directly from a single incoming input does not fully utilize the framework of ROS as there is no clear robot structure when all the devices are lumped into one node. Further, it was found that the sending/receiving of the message is difficult to replicate in Webots because modifying the included prewritten code to be compatible within Webots required inspection of each applicable script and replacement of device drivers with Webots functions.

In response to these issues another layer of abstraction could be introduced, dividing the message between pseudo devices, nodes within the ROS workspace that publish/subscribe to a single type of data replicating how a device would perform within ROS as if ROS was running on the Pi. Essentially a simulation of the previous configuration on a PC, it uses the ROS structure making it easier to link to the Webots simulation and provides a more basic arrangement of ROS for beginner students. See Figure 5.

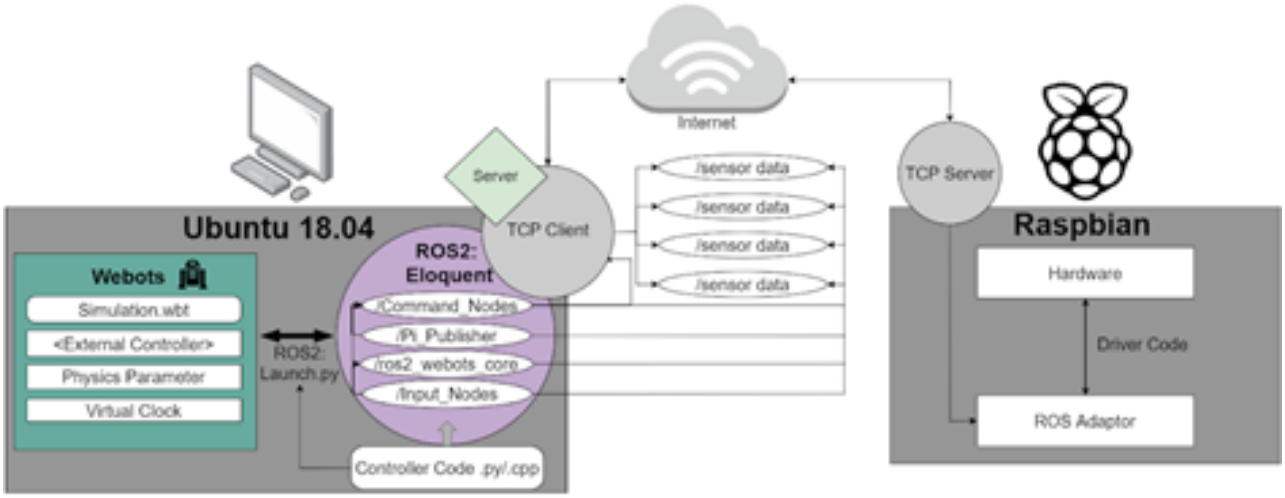

Figure 5: Configuration 2 details, a TCP client/server network, and significant nodes.

3.4 Installation Procedure

The installation of the software and necessary packages was found to be a complicated process as there is a diverse depth of documentation belonging to each package. It is anticipated that when distance learning, the confusion will be amplified by the nature of the situation. Further, students have varying degrees of experience with Linux and Raspberry Pi. A step-by-step instruction set for each platform used was created to provide clarity to this process. Additionally, a virtual machine image can be created with the necessary software preinstalled with optional preprogrammed examples.

4. Results

The resulting prototype of the simulation model in Webots is shown in Figure 6 based on a physical tank robot.

Figure 6. Prototype simulation model and physical robot.

While neither configuration (in figures 4 and 5) is necessarily better than the other for beginning level and distance learning education purposes, each has its advantageous. Further, the pairing of ROS2 and Webots is not limited to these configurations, packages, and communication protocols. There are many arrangements in connecting ROS and Webots for education not detailed in this report.

The specific connections constructed in this research were found to have various obstacles whose solutions are described in the following and reflected in Table II. Firstly, Webots runs most reliably with an NVIDIA or AMD OpenGL graphics adaptor. Neither Oracle nor VMware free versions, which were the virtual machine platforms utilized during this project, support Webots's graphics preference. This was found to have minimal effect on the performance of the simple simulations if the rendering options were adjusted. For heavier simulations, it would be advantageous to host directly on a Linux machine or create a remote connection to one. Secondly, the prevalent GPIO library for Raspberry Pi is no longer receiving support. Consequently, prewritten code cannot be reused within the ROS framework. There is a release which works on Ubuntu 20.04 but not for 18.04 which was used in this research. ROS2 releases a version compatible with Ubuntu 20.04 which can be used in substitution with little impact, or all GPIO and driver code can be written from scratch/using alternate libraries. Third, it was observed that ROS2 and Webots often have complications in connecting when installed separately. Using the webots_ros2_desktop package is a reliable way to smoothly use the two jointly.

TABLE II. Comparison of key elements in proposed configurations.

	Config 1: ROS on PI	Config2: ROS off Pi
Desktop/Laptop		
Run Ubuntu 18.04	x	x
Use Virtual Machine	x	x
Run ROS2 Eloquent	x	x
Webots Simulation Software	x	x
Handles Large ROS workspace		x
Intelligence Code		x
Sudo Peripheral Node Structure		x
TCP Network Socket Communication		x
ROS2 Network Communication	x	
Potential Multi Robot Network	x	x
Raspberry Pi		
Ubuntu Server for Pis	x	
ROS2 Eloquent	x	
Raspbian/Raspbian Based OS		x
Supporting GPIO Access Libraries		x
ROS Serial Communication Protocol	x	
Option for Onboard ROS intelligence	x	
Possibly Write Drive Code	x	
Webots		
webots ros2 Package	x	x
Desktop Suite Installation Option (Installs Webots with package)	x	x
Easily Modify Existing Robot Code		x
Accurately Model Exact Processes on Pi	x	
Speed-up/Slow-down Simulation	x	x

5. Conclusions

In this work, we have presented several approaches for introducing students to the field of robotics with an eye toward online distance education. A mature Introduction to Embedded Systems course was described, and in response to the COVID-19 pandemic the path taken to deploy a fully remote virtualized version of its core experiential platform (i.e. CyBot) was discussed. An exploratory look at several state of the art robot development technologies (ROS2, Webots, Raspberry Pi) was given for developing an approach to introduce students to robotics in a manner that is approachable for students, while not limiting those who want to explore advanced aspects of robotics. As part of this exploration, recommendations were given with respect to concerns to keep in mind, and mitigation advice was offered.

Acknowledgments

We appreciate the contributions of Maggie Heaslip to the summer undergraduate research project. This material is based upon work supported by the National Science Foundation (NSF) under awards DGE-1303279, EEC-1565130 and EEC-1623125. Any opinions, findings, and

conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

Batth, R., Nayyar, A., & Nagpal, A. (2018). Internet of Robotic Things: Driving Intelligent Robotics of Future - Concept, Architecture, Applications and Technologies. *2018 4th International Conference on Computing Sciences (ICCS)*, (pp. 151-160). doi:10.1109/ICCS.2018.00033

Beavis, P., Sardar, M., Sircin, L., Janack, G., Pack, D., Griffith, A., & Barrett, S. (2005). Using Robots to Teach Complex Real Time Embedded System Concepts. *Proc. 2005 ASEE Annual Conference.* Portland, Oregon. Retrieved from https://peer.asee.org/14719

Berry, C. (2010). Mobile Robotics: A Tool for Application Based Integration of Multidisciplinary Undergraduate Concepts and Research. *Proc. 2010 ASEE Annual Conference & Exposition.* Louisville, Kentucky. Retrieved from https://peer.asee.org/15642

Berry, C. A. (2017). Robotics education online flipping a traditional mobile robotics classroom. *Proc. 2017 IEEE Frontiers in Education Conference (FIE)* (pp. 1-6). IEEE. doi:10.1109/FIE.2017.8190719

Drushel, R., & Gallagher, J. (2008). The Virtual Classroom Environment of a WWW Based Autonomous Robotics Laboratory: Factors Affecting Student Participation, Communication, and Performance. *Proc. 2008 ASEE Annual Conference & Exposition.* Pittsburgh, Pennsylvania: ASEE Conferences. Retrieved from https://peer.asee.org/3923

Englberger, F., Latzel, T., & Sotiriadis, P. (2018). An Autonomous Robot for Embedded Systems and Robotics. *2018 12th European Workshop on Microelectronics Education (EWME)* (pp. 35-39). IEEE. doi:10.1109/EWME.2018.8629475

Fan, K.-Y. D., & Dimiduk, C. K. (2011). Using the Matlab-based iRobot create simulator to engage introductory computer programming students in program development and observing computational errors. *Proc. 2011 Frontiers in Education Conference (FIE)* (pp. S2G-1-S2G-6). IEEE. doi:10.1109/FIE.2011.6143104

Gennert, M. A., & Putnam, C. B. (2018). Robotics as an Undergraduate Major: 10 Years' Experience. *Proc. 2018 ASEE Annual Conference & Exposition.* Salt Lake City, Utah: ASEE Conferences. Retrieved from https://peer.asee.org/30943

Guo, Y., Zhang, S., FAIMBE, A. B., & Man, H. (2012). Teaching Micro-robots in Biomedical Applications: A Modified Challenge-based Pedagogy and Evaluations. *Proc. 2012 ASEE Annual Conference & Exposition.* San Antonio, Texas: ASEE Conferences. Retrieved from https://peer.asee.org/22008

Guyot, L., & Rohrer, F. (2011). Teaching robotics with an open curriculum based on the e-puck robot , simulations and competitions. Cyberbotics.

Hur, B., Goulart, A. E., Porter, L., Sarker, N., & Willey, M. (2020). Embedded System Education Curriculum Using TI SimpleLink Microcontrollers in Engineering Technology. *Embedded System Education Curriculum Using TI SimpleLink Microcontrollers in Engineering Technology.* Virtual ASEE Conference. doi:10.18260/1-2--34518

Lessard, R. A. (1999). Embedded Systems Course Focuses On Autonomous Robot Applications. *Proce. 1999 ASEE Annual Conference.* Charlotte, North Carolina: ASEE Conferences. Retrieved from https://peer.asee.org/7626

Liang-Yi Li, C.-W. C.-D. (2009). Researches on Using Robots in Education. In K. R. Chang M., *Learning by Playing. Game-based Education System Design and Development. Edutainment 2009. Lecture Notes in Computer Science, vol 5670* (Edutainment 2009 ed., Vols. Lecture Notes in Computer Science, vol 5670). Berlin, Heidelberg: Springer. Retrieved from https://doi.org/10.1007/978-3-642-03364-3_57

Meruje, M. &. (2018). *A Tutorial Introduction to IoT Design and Prototyping with Examples.* doi:10.1002/9781119456735.ch6

Micheal, O. (2004, 03). WebotsTM: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems, 1*. doi:10.5772/5618

Nakamoto, N., & Kobayashi, H. (2019). Development of an Open-source Educational and Research Platform for Autonomous Cars. *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society* (pp. 6871-6876). IEEE. doi:10.1109/IECON.2019.8926794

Pan, Y., Ma, X., Mu, C., An, H., & Chen, J. (2018). Design of Industrial Robot Sorting System with Visual Guidance Based on Webots. *2018 3rd International Conference on Computer and Communication Systems (ICCCS)* (pp. 516-521). IEEE. doi:10.1109/CCOMS.2018.8463315

Quigley, M., Conley, K., Gerkey , B., Faust, J., Foote, T., Jeremy, L., . . . Ng, A. (2009, 01). ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software, 3*.

Rover, D.T., Fila, N.D., Jones, P.H. and Mina, M. (2020). Introducing Autonomy in an Embedded Systems Course Project. To appear, *Proc. of the IEEE/ASEE Frontiers in Education Conference*, October 21-24, 2020.

Su, L. J. (2019). *Open-Source Robotics Projects.* Singapore: ABI Research.

Tobe, F. (2017, November 21). *30+2 research reports forecast significant growth for robot industry*. Retrieved from The Robot Report: https://www.therobotreport.com/302-research-reportsforecast-significant-growth-robot-industry/

Transfer to Your Own Robot - Webots documentation. (n.d.). Retrieved July 20, 2020, from https://cyberbotics.com/doc/guide/transfer-to-your-own-robot

Webots: Robot Simulator. (n.d.). Retrieved July 14, 2020, from https://cyberbotics.com/

Wilkerson P.E., S. A., Forsyth, J., Sperbeck, C., Jones, M., & Lynn, P. D. (2017). A Student Project using Robotic Operating System (ROS) for Undergraduate Research. *2017 ASEE Annual Conference & Exposition.* Columbus, Ohio: ASEE Conferences. doi:10.18260/1-2--27515