Promoting Open-source Hardware and Software Platforms in Mechatronics and Robotics Engineering Education

Dr. Nima Lotfi, Southern Illinois University, Edwardsville

Nima Lotfi received his B.S. degree in electrical engineering from Sahand University of Technology, Tabriz, Iran, in 2006, his M.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2010, and his Ph.D. degree in mechanical engineering from Missouri University of Science and Technology, Rolla, MO, USA, in 2016. He is currently an Assistant Professor with the Mechanical Engineering Department at Southern Illinois University Edwardsville, Edwardsville, IL, USA. His current research interests include characterization and electrochemical modeling of Li-ion batteries, traditional and electrochemical model-based Li-ion battery management system design, and real-world applications of control and estimation theory especially in alternative and renewable energy systems, mechatronics, robotics, and electrified and autonomous transportation. Dr. Lotfi is a member of the IEEE Control Systems Society and ASME Dynamic Systems and Control Division.

Mr. Kenechukwu Churchill Mbanisi, Worcester Polytechnic Institute

Kenechukwu C. Mbanisi received the B.Eng. degree in electrical and electronic engineering from Covenant University, Nigeria, in 2013, and the M.S. degree in robotics engineering from Worcester Polytechnic Institute (WPI), MA, USA in 2018. He is currently working towards the Ph.D. degree in robotics engineering from WPI, USA. His research interests include human motion modeling, planning and analysis, human-robot and human-machine interaction.

Dr. David M. Auslander, University of California, Berkeley

David M. Auslander is Professor of the Graduate School, Mechanical Engineering, University of California at Berkeley. His interests include mechatronics, real time software, and mechanical control. Current projects are building energy control, satellite attitude control, mechanical system simulation, and engineering curriculum. He consults in control and computer applications and legal matters. He was a co-founder of Berkeley Process Control, which sold mechanical control products. His education was at Cooper Union and MIT. He has awards from several engineering organizations.

Dr. Carlotta A Berry, Rose-Hulman Institute of Technology

Dr. Carlotta A. Berry is a professor in the department of Electrical and Computer Engineering at Rose-Hulman Institute of Technology. She is the director of the multidisciplinary minor in robotics and co-director of the Rose building undergraduate diversity scholarship and professional development program. She has been the President of the Technical Editor Board for the ASEE Computers in Education Journal since 2012. She is a member of ASEE, IEEE, NSBE, and Eta Kappa Nu.

Dr. Luis Alberto Rodriguez, Milwaukee School of Engineering

Dr. Luis A. Rodriguez is currently an assistant professor in the Mechanical Engineering Department at the Milwaukee School of Engineering (MSOE). He completed his doctoral training at the University of California-Irvine where he was a National Science Foundation Bridge to the Doctorate Fellow. He completed his master's degree at the University of Wisconsin-Madison where he was a GEM fellow and Graduate Engineering Research Scholar. He also holds a bachelor's degree from University of California San Diego. His interests include robot control, design of mechatronics systems, pneumatic actuation, motion planning and optimal control.

Dr. Majid Molki, Southern Illinois University Edwardsville

Majid Molki is a Distinguished Research Professor of Mechanical Engineering at Southern Illinois University Edwardsville. He received his Ph.D. in Mechanical Engineering from the University of Minnesota, Minneapolis. During the nearly four decades of academic work, he has explored many aspects of thermal-fluid sciences. He has extensive research records in experimental heat and mass transfer. His more recent activities are focused on computational modeling of viscous Newtonian and non-Newtonian fluids, turbulent flows, and targeted heat transfer augmentation using electrically-induced corona jet.

Promoting Open-source Software and Hardware Platforms in Mechatronics and Robotics Engineering Education

Abstract

The evolution of Mechatronics and Robotics Engineering (MRE) has enabled numerous technological advancements since the early 20th century. Professionals in this field are reshaping the world by designing smart and autonomous systems aiming to improve human well-being. Recognizing the need for preparing highly-educated MRE professionals, many universities and colleges are adopting MRE as a distinct degree program. One of the cornerstones of MRE education is laboratory- and project-based learning to provide a hands-on and engaging experience for the students. To this end, numerous software and hardware platforms have been developed and utilized in MRE courses and laboratories. Commercial products can provide a rich hands-on experience for the students, but they can be cost-prohibitive. On the other hand, open-source platforms are low-cost alternatives to their commercial counterparts and are being increasingly used in industry. Developing open-source laboratory platforms will be a more feasible option for a wider range of institutions and would enable familiarizing the students with recent technological trends in industry and exposing them to the development details of a real-world system. However, adoption of open-source platforms in MRE courses can be lengthy and time consuming. Educators who wish to utilize such systems typically lack the expertise in all aspects of their implementation which can make them difficult to troubleshoot. Debugging open-source systems can also be challenging because most of the troubleshooting is done through forum discussions which appear to be very noisy and unfocused. The flip side of this chaotic nature of the open-source world is that there is a vast amount of information available, including tutorials, examples, and commentary and, with some focused searching, debugging and usage questions can often get answered. There is also a disconnect between the forum participants, typically computer scientists and hobbyists, and MRE educators and students. Finally, the available resources and documentation for utilizing open-source platforms in MRE education are insufficient and incomprehensive. Therefore, the main goal of this paper is to increase awareness and familiarity with the use of open-source software and hardware packages in MRE education and practice towards accelerating their adoption. To this end, open-source software packages such as Python, GNU Octave, OpenFOAM, Java, Modelica, Gazebo, SPICE, Scilab, and Gnuplot, which have the potential to be useful in the modeling and analysis of MRE systems are introduced. Furthermore, low-cost and powerful open-source hardware packages such as Arduino, Raspberry Pi, and BeagleBone which can be used as the main processing unit for data acquisition and control implementation in a wide range of MRE systems are reviewed and their limitations and potentials are investigated. This paper provides a valuable resource for MRE students and faculty who would like to utilize open-source hardware and software platforms in their education and research.

1 Introduction

The field of Mechatronics and Robotics Engineering (MRE), nowadays, involves a synergistic integration of precision mechanical engineering with electronics and intelligent computer control in the design of manufacturing processes and smart products. In recent years, MRE has experienced tremendous, dynamic growth owing to advances in integrated circuits and electronics, embedded systems and computers, networks, and intelligent systems, as well as democratization of access through open-source hardware and software and the Maker Movement. MRE professionals are shaping the world by designing smart and autonomous systems and processes that will improve human life and welfare. Application areas in which MRE is increasingly being used and needed include consumer electronics, smart home appliances, building automation, medical and healthcare applications, heavy machinery, manufacturing, autonomous vehicles in ground transportation and aviation industries, and industrial and space robotics.

MRE education requires an interdisciplinary knowledge of mechanical, electrical, computer, software, and systems engineering to oversee the entire design and development process. To prepare the next generation of engineers for this fast-paced field and to fill the skill-gap that the traditional engineering graduates exhibit, many universities and colleges have introduced MRE courses, minors, and degree programs. However, there is not a well-defined and unified framework for such degree programs, which can cause confusion and ambiguity among instructors and future employers. To overcome this challenge, the authors, with financial support from NSF, have been engaged in holding several workshops on the Future of Mechatronics and Robotics Engineering Education. These workshops have brought together more than 150 faculty, students, and industry professionals in the MRE field to share broad success stories; to develop concept inventories for MRE curricula and courses; to identify thought leaders; to learn the recent trends in industry; and to develop a roadmap for MRE education. These workshops have aimed to contribute to the quality of MRE education and increase adoption to prepare individuals with a blend of theoretical knowledge and practical hands-on learning. These future Mechatronics and Robotics engineers, with technical breadth and depth, will be uniquely prepared to develop the systems that define the future of work at the human-technology frontier.

The majority of technical MRE courses include a heavy theoretical focus which could be overwhelming, especially for undergraduate students. However, in the past decades, numerous efforts have been made towards developing software and hardware packages to provide a hands-on and engaging experience for the students through laboratory and project-based learning paradigms. Such an experience could give students a deeper understanding of the core MRE concepts as well as an opportunity to practice the application of those concepts – linking knowledge and real-world skills. Students who demonstrate this understanding can confidently study real-world global challenges, devise multiple solutions to these problems, evaluate the feasibility of the various solutions based on the qualitative and quantitative criteria and constraints, and implement the optimal solution while considering the broader societal impacts.

Commercial products from companies such as MathWorks, National Instruments, Texas Instruments, and Quanser can provide a rich hands-on experience for the students in the field of MRE. The procurement and acquisition costs, however, can be a big hurdle for educational institutions which lack the necessary financial infrastructure. On the other hand, open-source

platforms are low-cost alternatives to their commercial counterparts and are also being increasingly used in industry.

The open-source community for technology took off in the early 1980s and has continued to grow ever since. The "open-source way", first applied to software development, distribution, and maintenance, was motivated by the need for better collaboration amongst developers, improved accessibility and reliability of software through this network of contributors. In line with this movement, open-source software (OSS) was defined as software released under a license which grants users the right to study, use, modify, and distribute the source code to anyone and for any purpose [1]. Software such as Linux operating system and Apache Web server are some of the early developments of the community. Today, a plethora of high-quality open-source alternatives to proprietary software are available for various applications and have gained widespread adoption in academia and education. This adoption in academia is motivated by flexibility and affordability which open-source software provide to educators and institutions, who are oftentimes unable to afford the expensive licensing fees of commercial/proprietary software options. Common examples of OSS frequently adopted by academia for modeling, computation and analysis are Python, GNU Octave, Scilab, Modelica, C, C++, Java, Real-time Linux, Gazebo.

The steady adoption and success of OSS sparked the initiation and rise of another open-source movement, open-source hardware (OSH). Though much of the efforts in standardization, definition and certifications started in the late 1990s, it was not until the mid-2000s that the movement truly took off with the creation of projects and companies such as Arduino, AdaFruit, OpenCores and SparkFun [2]. OSH "refers to the design specifications of a physical object which are licensed in such a way that said object can be studied, modified, created, and distributed by anyone." [3] To enable accessibility and reproducibility, OSH generally follow standard design processes with open-source tools and content, as well as readily available components [4]. The growth of both OSS and OSH has led to the development of the global Maker community, a community of enthusiasts and hobbyists interested in designing and prototyping projects of various kinds. As with OSS, the cost of proprietary hardware equipment for education motivated the widespread adoption of OSH in academia.

Today, classrooms all over the world, from K-12 to college, use products such as Arduino and Raspberry Pi to facilitate classroom, laboratory and design project experiences in a wide range of disciplines such as textiles [5], chemical instrumentation [6], physics experiments [7], photovoltaics [8] and additive manufacturing [9]. But by far, the most prevalent application of open-source platforms for education has been in the electronics and computer engineering/programming disciplines, specifically in controls, automation, embedded systems [4], which are the foundational disciplines of MRE. From the OSH standpoint, three main low-cost prototyping platforms, which tower above others in adoption, are the Arduino, Raspberry Pi and BeagleBone. The Arduino, a microcontroller-based board, was a pioneering OSH platform which has found tremendous application in MRE-related disciplines such as in teaching control systems [10-12], embedded systems [13], mechanical design [14], and the list goes on. Its ease of use, intuitive IDE, and extensive online community of contributors and supporters make it easy for both educators and students to adopt in the classroom. Raspberry Pi, since its release in 2012, has grown in popularity as a full-fledged computer in a credit-card size form factor. Its superior processing power, networking capabilities, and versatility make it better suited for more

comprehensive embedded systems class or project experiences than Arduino [15]. On the other hand, the BeagleBone combines the easy-to-interface functionality of the Arduino and the power and versatility of a Linux operating system (just as Raspberry Pi). These features have driven its application in embedded systems and robotics [15-17]. When high-level computational power is required for advanced robotics, computer vision and machine learning applications, OSH platforms such as the Nvidia Jetson microcomputers are used [18].

Developing open-source platforms for MRE education will not only be a more feasible option for a wider range of institutions but can also prepare the students with recent technological trends in industry. Utilizing open-source platforms can expose the students to the development details of a real-world system and therefore, can provide a deeper learning experience. However, the entire development process using open-source platforms and its troubleshooting can be very lengthy and time consuming. Students with good hands-on skills are typically needed in order to develop such platforms; furthermore, the educators who wish to utilize such systems typically lack the expertise in all aspects of their implementation. Debugging open-source systems can also be challenging as majority of the troubleshooting is done through forum discussions which appear to be very noisy and unfocused. There is also a disconnect between the forum participants, typically computer scientists and hobbyists, and MRE educators. Finally, the available documentation for utilizing open-source platforms for the design, analysis, and real-time implementation of MRE systems are incomprehensive and insufficient.

The main goal of this paper is to increase awareness and familiarity with the use of open-source software and hardware packages in MRE education and practice towards accelerating their adoption. To this end, Section 2 will provide an overview of the available open-source software packages that can be used in the modeling and analysis of MRE systems. Real-time data acquisition and control will be studied in Section 3 and hardware platforms that can be used for these purposes will be reviewed. Finally, Section 4 provides a summary of the paper along with a roadmap for further integration and utilization of the open-source platforms in MRE education.

2 Open-source Software Packages for Modeling and Analysis of MRE Systems

Modeling refers to mathematical representation of certain characteristics of the system which are of interest for a specific application. It is widely used in the analysis and control of MRE systems. The choice of the model mainly depends on the application and available computational resources. The models used for the purpose of analyzing the system behavior are typically high-fidelity physics-based models, as the accuracy of the analysis is highly dependent on the model fidelity. In applications involving large datasets or where the underlying physics is unknown and difficult to characterize, high-order empirical models could also be utilized. In the context of control design, however, lower-order compact and semi-empirical models are commonly used. Specially, the disturbance rejection nature of the feedback controllers could eliminate the need for highly accurate models. Furthermore, lack of computational resources is another reason for using lower-order models in the control design. As will be outlined later in the section, with the advancements in microcontrollers and microcomputers, this challenge is becoming less significant.

Despite the type of the model, MATLAB and Simulink have been the main commercial tools for modeling and analysis of MRE systems in the past few decades. However, the following open-source software platforms are also increasingly being developed and improved to undertake various modeling tasks.

2.1.1 Python

In the early days of mathematical/scientific computing, compiling a program before running was a very laborious and time-consuming process. Thus, interpreted languages (e.g., Basic) were introduced. They traded off compile time (no compilation needed) against much slower execution time. In the modern computing world, however, that distinction has largely disappeared -- typical Java or C++ programs, for example, compile in a fraction of a second.

Python is a general-purpose and interpreted programming language, first developed by Guido van Rossum and released in 1991. In order to make it freely accessible to everyone, it is developed under an open-source license. Python Software Foundation, which is responsible for administering Python's license, was formed "to promote, protect, and advance the Python programming language, and to support and facilitate the growth of a diverse and international community of Python programmers" [19]. Due to its high-level and open-source nature, large community of developers in Python, an abundance of packages developed for Python [20], and easy integration with other programming languages, Python has gained popularity in various applications and disciplines and has turned into an industry standard. PyCon is an annual gathering for Python developers and users that provides different tutorials, a conference, development sprints, summits, and a job fair [21].

Considering the wide range of Python applications, there are a lot of resources for getting started with Python. The website [19] provides a rich repository of documentation about Python installation, learning materials, news, and events. Comprehensive lists of books about Python and its applications can be found at [22-23]. Furthermore, there are several online Python courses on MITOpenCourseWare [24], Coursera [25], Future Learn [26], etc.

In the past decade, the Python community has developed numerous packages for mathematics, science, and engineering applications. These packages enable the implementation of optimization, linear algebra, ODE solvers, interpolation, FFT, and signal processing [27]. The following is a list of some of these libraries

- SciPy: A comprehensive library with sub-modules for various mathematical and scientific operations such as Fast Fourier Transform, interpolation, numerical integration, linear algebra, file input/output, optimization and curve-fitting, statistics, and signal processing.
- NumPy: Offers computational capabilities similar to MATLAB which also overlaps with some of SciPy's functionalities. It is best suited for fast array operations. MATLAB users who are wishing to learn more about NumPy can refer to [28] for a detailed comparison between MATLAB and NumPy.
- Matplotlib: Enables high-quality 2D plotting. It can be used to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc.

- SymPy: Used for symbolic programming. It is incorporated into a wide range of other libraries such as symbolic statistical modeling, Multibody dynamics, linear circuit analysis, etc.
- IPython: Provides an interactive shell similar to MATLAB's workspace which is ideal for troubleshooting, interactive data visualization, use of GUI toolkits, and parallel computing.
- pandas and Sickit-learn: Offer an extensive suite of tools for predictive data analysis, modeling, and machine learning.

Some of the applications of these packages, specifically data analysis and visualization, audio and digital signal processing, image and video processing, and control implementation can be found in the online course, Python for Scientists and Engineers [29].

In addition to the above general-purpose libraries which can be used for dynamic system modeling, Python has an exclusive library called Python Control Systems Library [30]. This library provides functions and classes in order to systematically analyze and design feedback control systems. The toolbox is also equipped with MATLAB compatibility module to enable the utilization of some of the commands in MATLAB Control Systems toolbox. Specific to modeling, this toolbox can be used to represent LTI systems in transfer function, state-space, or frequency-response data forms and simulate them in time and frequency domains. Implementation details of the functions and classes in Python Control Systems Library can be found at [30]. The combination of these packages can pose Python as a powerful tool in the analysis and modeling of dynamic systems. However, a constant challenge for MRE faculty and students wishing to utilize Python Control Systems library is to get over the initial installation and setup hurdles. Furthermore, there is still a need for documentation and troubleshooting supported by the MRE community and use case studies to ease the integration of Python into modeling courses. Finally, Python programming language, and specifically, Python Control Systems Library, can also be used for the analysis and design of control systems. In addition to enabling model representation and simulation in state-space and frequency domains, this toolbox can be used for model order reduction and block diagram algebra. Furthermore, it provides the following useful features [30]:

- Control analysis: stability, reachability, observability, stability margins
- Control design: eigenvalue placement, LQR, H_2, H_{inf}
- Estimator design: linear quadratic estimator (Kalman filter)

In modernizing the Python language over a decade ago, Python 3 was introduced. Although recognizably Python, it has some incompatibilities with Python 2. The official Python 2 support is ending this year, however, there are still libraries that have not been upgraded, complicating the move to Python 3.

2.1.2 GNU Octave

GNU Octave [31] is a high-level programing language for numerical computations of linear and nonlinear problems. Octave is an open-source software with a syntax very similar to and mostly compatible with MATLAB. Extensive tools available in Octave make it possible to solve a variety of engineering problems involving linear and nonlinear equations and ordinary differential equations. Octave with built-in plotting and visualization capabilities and compatibility with

MATLAB scripts can be used for modeling and control design in MRE systems. While Octave can use some MATLAB toolkits, it also has its specifically-designed libraries [31].

2.1.3 OpenFOAM

OpenFOAM [32] is a free open-source software for Computational Fluid Dynamics (CFD). With an extensive range of features, this software can solve complex fluid flow problems involving heat transfer, combustion, turbulence, multi-phase flows, and many more. Despite its focus on CFD, OpenFOAM is capable of incorporating solid mechanics and electromagnetics. This open-source software makes it possible to have access to the source code and therefore, is an ideal tool for teaching courses such as CFD. It is best suited as a tool for graduate courses so that the students can develop their computational model later in their research.

2.1.4 Java

While Java is not the first programming language environment that comes to mind in a mechatronics/robotics context, it has many properties that make it a very useful tool. The first of these is the language itself. Java is fully object-oriented, has a relatively clean syntax, executes quickly, and operates with no changes (usually) across different computers and operating systems. It can be applied to mathematically based problems (for example, dynamic simulation) as well as used for direct control of physical systems.

Java compiles to an abstract machine code that executes via a Java Virtual Machine (JVM). Thus, a compiled Java program can execute on any system for which a JVM exists. This guarantees that basic properties such as how integer and real numbers are stored, precision associated with different data types, etc., will not change from one environment to another.

Java is open-source. The Java Development Kit (JDK, compiler, linker, etc.) is available from [33]. Commercially licensed versions are also available from Oracle. The two most popular development environments (IDE) are also open-source, Netbeans [34] and Eclipse [35].

While the Java language itself provides very effective ways to organize complex problems, solving them often requires additional numerical math tools. For a wide range of problems, the appropriate tools are available through the open-source Apache Commons Math Library [36] or its successor, the Hipparchus Math Library [37]. There are also a number of smaller, more specialized libraries that can be found by searching.

2.1.5 Modelica

“The Modelica Language is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.” [38]

Modelica is of particular interest to the mechatronics/robotics community because it provides dynamic simulation of three-dimensional and constrained mechanical systems and energetically correct interactions between the mechanical systems and electrical systems, hydraulic, pneumatic,

etc. It does this through an underlying mathematical base that is capable of solving differential-algebraic equations (DAEs) and an underlying computing syntax based on equations rather than algorithmic "store-into" statements. Most users do not interact with Modelica at that level, but rather make use of an extensive set of libraries for various physical domains combined with a graphical user interface.

Systems with constraints are known as "acausal". For a system defined by a set of N first-order, nonlinear differential equations, constraints add algebraic equations to the system equations. Acausal systems have fewer than N independent initial conditions because the algebraic constraints reduce the effective order of the system. In mechatronics, almost all mechanical systems of any interest have constraints which is one reason Modelica is so valuable.

The Modelica language is defined by an open-source specification maintained by the Modelica Association [38]. The actual implementation is left to any organizations that wish to, which includes both commercial and open-source entities. The most important open-source implementation is OpenModelica [39]. Jmodelica [40] used to be open-source but no longer is.

2.1.6 Gazebo

Gazebo is an open-source 3D dynamic simulator widely used in robotics research and education. The software provides a framework for realistic simulation of multiple robots dynamically interacting with each other in complex environments and scenarios [41, 42]. Gazebo comprises a suite of features which include high-performance physics engines (e.g. Open Dynamics Engine, Bullet, etc.), high-fidelity graphical rendering of environments, and ability to generate realistic sensory data from the environment [41]. These features allow users to create very realistic simulated environments and scenarios to learn to operate and test their robot designs and control algorithms without the risk of damaging their physical robots. Users can either leverage the rich library of existing robot models available with the software (ranging from mobile aerial and ground robots to state-of-the-art humanoid robots) or design their own robot models from scratch using the Unified Robot Description Format (URDF).

2.1.7 SPICE

SPICE (Simulation Program with Integrated Circuit Emphasis) is an open-source, general-purpose circuit simulator used in the design process of electronic circuits to evaluate circuit models and predict their behavior. The software allows users to model linear and nonlinear circuits using a built-in library of electronic components such as passive elements (resistors, capacitors, inductors, diodes etc.), semiconductor devices, dependent and independent DC and AC sources, transmission lines, and many more. The circuit model, including all components, their connections and operating values, can be designed in a text editor (as a text file called a netlist) or using a graphical schematic editor. SPICE enables various types of circuit analysis to be performed such as DC, transient, AC small-signal, pole-zero, small-signal distortion analyses [43, 44]. Since its original development in the mid-70s by researchers at the University of California, Berkeley, several variations have been developed such as Ngspice, pSPICE, and LTspice. SPICE, and its variants, have been used extensively in electrical/electronic engineering courses and labs in teaching students about electronic circuit design [45, 46].

2.1.8 Scilab

Scilab [47] is a free and open-source high-level numerically-oriented programming language that is compatible with Windows, Mac OS, and Linux. It is an open-source alternative to MATLAB and provides many mathematical functions for numerical analysis. It also features a rich graphical environment for data visualization and application development, an interpretative language environment for quick algorithmic development, a powerful hybrid dynamic systems modeler and simulator named Xcos, and many toolboxes maintained by Scilab's ATOMS package repository. Xcos is a graphical programming environment similar in style to Simulink that is based on graphical block diagramming and comes with many block libraries. Xcos can be used to not only simulate dynamic systems and models but also with additional toolboxes can be used to control hardware and instrumentation such as the Arduino microcontroller and many National Instruments devices that interface with LabVIEW. Scilab also features many toolboxes for MRE system design such as Image Processing and Computer vision, Control Design, Optimization, Signal Processing, and a Robot toolbox for modeling, controlling and studying robot manipulators. Scilab also provides a rich repository of tutorials to help new users learn the environment and for more advanced users it has the capability to dynamically compile and link other languages such as Fortran and C.

2.1.9 Gnuplot

Although many analysis and simulation software packages include some form of plotting software, there are occasions when an external plotting package can be very useful. Gnuplot [48] is an open-source, multi-platform package that can produce 2D and 3D publication-quality graphics in a variety of media formats. Gnuplot has an extensive command-line interface but is probably used more often as a graphics scripting language to produce plots from data in a text file. In this mode, it works very well with software that produces output to a file and Gnuplot produces higher quality output than internal graphics, there is not a good internal graphics package, or simulation runs, for example, take a long time so if a graph needs to be redrawn having the output data the simulation does not have to be redone because the data has been saved to a file. It is also useful for maintaining a uniform appearance if data is coming from several sources.

3 Data Acquisition and Control Implementation

Data acquisition and control implementation are the two most important pillars of MRE systems. These functionalities are usually accomplished by a computing unit, or "brain", of the system, which could be in the form of a computer or a microcontroller. This unit is responsible for acquiring measurements from the system's environment, making decisions based on the sensors' measurements and also, the control goals, and implementing the decision using system actuators. Successful integration of a computing unit into an MRE system requires an in-depth knowledge of the data acquisition, interface theory, programming, and control implementation. In this section, three open-source hardware platforms, which are commonly encountered in MRE systems, will be introduced and their features, capabilities, and limitations will be reviewed.

3.1.1 Arduino Microcontrollers

Arduino microcontrollers are open-source electronics platforms that are easy to use for interfacing with hardware and software [9]. They are ideal as the brain of MRE systems because of the diversity of products and large online resources including tutorials and examples for integration with a multitude of peripherals and sensors. It is well-suited for fast prototyping for expert and novice electronics and programming users. The Arduino can be programmed to read analog and digital inputs such as buttons, sonar, infrared sensors, photoresistors, color sensors, Pixy cameras, etc. It can also be programmed to send digital and analog (PWM) outputs to drive motors, turn on LEDs, open/close a gripper, sound a buzzer, write text output to an LCD, and do many more functionalities. More resources are available on the Arduino forum, Arduino project hub as well as websites such as Instructables, SparkFun, AdaFruit, HowToMechatronics and YouTube.

Some of the benefits of Arduino over other systems include the cost, flexibility, programming environment, and open-source features. The Arduino is relatively inexpensive with some boards costing as little as $50, with many clones less than that. The Arduino is flexible because it can be used with Windows, Mac, OSX, and Linux operating systems. An Arduino program is a sketch that can be expanded with C++ libraries or extended to the AVR C programming language. The Arduino software is easy to use for beginners as well as experts because there are multiple alternatives for coding including the Arduino IDE, graphical (Ardublock, Scratch for Arduino), web-based (Arduino Create), and Atmel Studio [50]. The Arduino board can also be extended or cloned since the plans are published under a Creative Commons license.

Arduino products include the traditional microcontroller board, a smaller board called a module, shields that plug into the board to add on more features, as well as kits. The Starter Kit includes the microcontroller as well as electronic components and a project book. The entry level boards include the Arduino Uno, Arduino Leonard, and the Arduino 101. The entry level modules include the Arduino Esplora, Arduino Micro, and Arduino Nano. There are also products with more enhanced features such as Arduino MEGA 2560 and Arduino DUE. There are also IoT products such as Arduino Yun and products for wearable electronics.

The most popular Arduino boards for robotics and mechatronics projects are the Arduino Uno, Arduino MEGA 2560 and Arduino Nano, due to the cost, size, I/O, and memory. The Arduino Uno has 32 kB flash memory, 2 kB SRAM, Analog inputs, 14 Digital I/O, 6 PWM, and 1 UART. A smaller and cheaper option is the Arduino Micro with 32 kB flash memory, 2.5 kB SRAM, 2 analog inputs, 20 digital I/O, 7 PWM, and 1 UART. If the project requires more I/O, the Arduino MEGA 2560 is a great option because it has 16 analog inputs, 54 digital I/O, 15 PWM, 256 kB flash memory, 8 kB SRAM, and 4 UART, although the cost is about double the Uno.

3.1.2 Raspberry Pi

Raspberry Pi is a generic name for a family of credit card-sized single-board computers, initially developed by Raspberry Pi Foundation [51], in response to the increasing demand in educational institutions for integrating computing as a part of their curricula. Due to its low-cost, flexibility, and versatility, Raspberry Pi has gained popularity at various educational levels from K-12 to higher-education institutions. It is also increasingly being used in hobby and industrial projects such as robotics, drones, and automation. In March 2018 and almost 6 years after its initial release, more than 19 million Raspberry Pi units were sold globally [51].

Raspbian, a Linux distribution, is the recommended operating system for Raspberry Pi but other third-party operating systems have also been developed for it [51]. Similarly, while the main programming language for Raspberry Pi is Python, other languages could also be used and have been developed on Raspberry Pi. Some of the features of Raspberry Pi 4 Model B, the latest model released on June 2019, which make it attractive for MRE education include

- **Size:** 85.60 mm × 56.5 mm × 17 mm
- **Weight:** 46 g
- **Cost:** $35/$45/$55 (For 1, 2, or 4 GB of SDRAM, respectively)
- **Processing:** 1.5 GHz 64-bit quad core ARM Cortex-A72 processor and Broadcom VideoCore VI @500 MHz GPU
- **Connectivity:** On-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet, two USB 2.0 ports, two USB 3.0 ports
- **On-board storage:** MicroSDHC slot
- **Input/output:** Camera interface, 2 × HDMI ports and audio input and output

In addition to these features, Raspberry Pi also includes a header of 40 General Purpose Input Output (GPIO) pins [51]. While the majority of these pins can be used as digital I/O pins, some of the specialized ones can facilitate communication protocols such as SPI, I2C, and serial. Furthermore, all of these pins can be used to generate software PWM while hardware PWM is only available on 4 pins (GPIO12, GPIO13, GPIO18, GPIO19). Some of the GPIO header pins can also be used to connect to Raspberry Pi HATs (Hardware Attached on Top), which are special boards developed to connect Raspberry Pi to common MRE devices such as motor controllers, displays, and touch sensing. A wide selection of Raspberry Pi HATs can be found on [52].

Considering the numerous capabilities of Raspberry Pi, it can be used to teach a variety of topics including: Introduction to Linux, Python, interface to sensors and actuators, image processing, robotics, and controls. As mentioned earlier, Raspberry Pi GPIO pins can only be used to interface to digital sensors and actuators. Analog actuators can be connected using the PWM functionality of the GPIO pins. In order to acquire measurements from analog sensors, specialized ADC chips such as ADS1115 and MCP3008 need to be used. Another alternative is to combine Raspberry Pi with a low-level microcontroller such as Arduino, however, care should be taken to ensure safety due to different logic levels used in these systems (0-5V in Arduino versus 0-3.3 V in Raspberry Pi). The use of a logic level converter can overcome this obstacle. The combination of Arduino and Raspberry Pi is a very powerful tool to be used in MRE systems. In the end, there is a rich library of project ideas using Raspberry Pi [51]. Specific MRE-related laboratory ideas can also be found at [53].

3.1.3 BeagleBone

BeagleBone boards are low-cost and community-supported development platforms. BeagleBone boards are comparable in size with the Arduino microcontroller; however, they differ in performance. A BeagleBone Black, for instance, is about 40 times faster than the Arduino UNO, it has about 128,000 times more RAM, and runs off a microSD flash card. BeagleBone boards are also capable of running a Linux operating system and connect to the internet without any additional

hardware add-ons or shields. Having the ability to run Linux onboard allows greater flexibility and the ability to use open-source software for the development of MRE systems. Additionally, similar to the Arduino boards, the BeagleBone boards are equipped with ADCs to interface with sensors, provide digital I/O, and PWM pins. Beaglebone boards are also able to extend their functionality by using hardware "capes", similar to Arduino shields or Raspberry Pi HATs. The BeagleBone website [54] also supports new users with a Learn support page, Book reference page, Discussion forum, and Project page among many other resources.

3.1.4 ROS

The Robot Operating System (ROS) is an open-source, middleware system framework for developing robot software. As a middleware, it serves as the nexus between the operating system (supports Linux Ubuntu) and the application programs operating the robots. ROS is particularly relevant in robotics because typical robot systems are complex and distributed, having multiple components (sensors, actuators, etc.) with different functionalities, possibly running on multiple computers. ROS serves as the framework which enables the modularization of the robot system functionality into "nodes" and then handles the communication between these respective system "nodes" [55]. In this way, ROS allows developers to separate their code base into software packages consisting of multiple single functionality programs, referred to as nodes. This enables reusability of robotics software and has led to the extensive library of ROS packages developed by the open-source community for a myriad of functionalities such as motion planning, vision and perception, control, etc [56]. ROS is designed to be language-agnostic, which allows users to implement software in any programming language, although the main languages used by the community are C++ and Python. Although ROS is not a hardware platform, its widespread use in real-time implementation and control of robotics is the main reason it is reviewed in this section. Although ROS can be implemented on single-board computers such as Raspberry Pi, it typically requires a more powerful computer.

4 Summary, Conclusions, and Future Work

Project-based learning through utilization of various software and hardware packages can provide MRE students with the necessary hands-on skills they would need to be competitive in the growing job market. Furthermore, such projects can help students better understand the theoretical concepts covered in the classroom. Despite their importance, the acquisition costs for commercial educational products can be prohibitive for institutions which lack the necessary infrastructure. Open-source platforms, on the other hand, can be suitable alternatives for their commercial counterparts and have the advantage that they allow greater flexibility and customization. To this end, this paper provides an overview of some of the open-source software and hardware platforms which have proven effective in MRE education and have wide community support. These platforms can complement various courses in MRE curricula, especially the core concepts such as modeling, analysis, and real-time implementation and control. Therefore, this paper can be very beneficial for MRE students and faculty who wish to utilize open-source platforms in their teaching and research. Future work includes creating a database of real-world examples and case studies of the use of these packages in various MRE courses. Finally, tutorials and documentation developed by MRE faculty can significantly help with widespread use and adoption of open-source platforms in higher education institutions.

References

[1] Laurent, A. M. S. (2004). Understanding open source and free software licensing: guide to navigating licensing issues in existing & new software. "O'Reilly Media, Inc.".

[2] Open Source Hardware Association (OSHWA). Brief History of Open Source Hardware: Organizations and Definitions. https://www.oshwa.org/research/brief-history-of-open-source-hardware-organizations-and-definitions/ [accessed December 2019]

[3] OpenSource.com. What are Open Hardware. https://opensource.com/resources/what-open-hardware [accessed December 2019]

[4] Heradio, R., Chacon, J., Vargas, H., Galan, D., Saenz, J., De La Torre, L., and Dormido, S. (2018). Open-Source Hardware in Education: A Systematic Mapping Study. IEEE Access, 6, 72094-72103.

[5] Buechley, L., Eisenberg, M., and Elumeze, N. (2007). Towards a curriculum for electronic textiles in the high school classroom. *ACM SIGCSE Bulletin*, 39(3), 28-32.

[6] Grinias, J. P., Whitfield, J. T., Guetschow, E. D., and Kennedy, R. T. (2016). An inexpensive, open-source USB Arduino data acquisition device for chemical instrumentation.

[7] Huang, B. (2015). Open-source hardware–microcontrollers and physics education–integrating diy sensors and data acquisition with arduino. In *the American Society for Engineering Education Annual Conference* (Vol. 26, pp. 1-26).

[8] Zachariadou, K., Yiasemides, K., and Trougkakos, N. (2012). A low-cost computer-controlled Arduino-based educational laboratory system for teaching the fundamentals of photovoltaic cells. *European Journal of Physics*, 33(6), 1599.

[9] Schelly, C., Anzalone, G., Wijnen, B., and Pearce, J. M. (2015). Open-source 3-D printing technologies for education: Bringing additive manufacturing to the classroom. *Journal of Visual Languages & Computing*, 28, 226-237.

[10] Hopkins, M. A., and Kibbe, A. M. (2014). Open-source hardware in controls education. *The ASEE Computers in Education (CoED) Journal*, 5(4), 62.

[11] Reguera, P., García, D., Domínguez, M., Prada, M. A., and Alonso, S. (2015). A low-cost open source hardware in control education. case study: Arduino-feedback ms-150. *IFAC-PapersOnLine*, 48(29), 117-122.

[12] Johne'M, P., and Canfield, S. L. Work-in-Progress: Using Hardware-based Programming Experiences to Enhance Student Learning in a Senior Feedback Controls Lecture Course. *age*, 23, 1.

[13] Jamieson, P. (2011). Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)* (p. 1). The Steering

Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[14] Bedillion, M. D., & Muci-Kuchler, K. H., and Nikshi, W. M. (2018), *An Arduino-Based Hardware Platform for a Mechanical Engineering Sophomore Design Course* Paper presented at 2018 ASEE Annual Conference & Exposition , Salt Lake City, Utah. https://peer.asee.org/29774

[15] Jamieson, P., and Herdtner, J. (2015, October). More missing the Boat—Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them. In *2015 IEEE Frontiers in Education Conference (FIE)* (pp. 1-6). IEEE.

[16] Bewley, T., Strawson, J., Ostovari, S., and Briggs, H. C. (2015). Leveraging Open Standards and Credit-Card-Sized Linux Computers in Embedded Control & Robotics Education. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (p. 0801).

[17] Betthauser, J., Benavides, D., Schornick, J., O'Hara, N., Patel, J., Cole, J., and Lobaton, E. (2014, April). WolfBot: A distributed mobile sensing platform for research and education. In *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education* (pp. 1-8). IEEE.

[18] Lotfi, N. A Multidisciplinary Course and the Corresponding Laboratory Platform Development for Teaching the Fundamentals of Advanced Autonomous Vehicles.

[19] Python.org. https://www.python.org/ [accessed January 19, 2020]

[20] STXNEXT Python Powerhouse. The Most Popular Python Scientific Libraries. https://stxnext.com/blog/2017/04/12/most-popular-python-scientific-libraries/?utm_campaign=Pat&utm_medium=Social&utm_source=Quora

[21] PyCon.org. https://us.pycon.org/2020/ [accessed January 19, 2020]

[22] Importpython.com. Books. https://importpython.com/books/ [accessed January 19, 2020]

[23] The Hitchhiker's Guide to Python. Learning Python. https://docs.python-guide.org/intro/learning/ [accessed January 19, 2020]

[24] MIT Open Courseware. https://ocw.mit.edu/index.htm [accessed January 19, 2020]

[25] Coursera.org. https://www.coursera.org/ [accessed January 19, 2020]

[26] Future Learn. https://www.futurelearn.com/ [accessed January 19, 2020]

[27] SciPy.org. https://www.scipy.org/ [accessed January 19, 2020]

[28] NumPy.org. NumPy for Matlab users. https://numpy.org/devdocs/user/numpy-for-matlab-users.html [accessed January 19, 2020]

[29] Python for Scientists and Engineers. https://www.pythonforengineers.com/python-for-scientists-and-engineers/ [accessed January 19, 2020]

[30] Python Control Systems Library. https://python-control.readthedocs.io/en/0.8.2/intro.html [accessed January 19, 2020]

[31] GNU Octave. https://www.gnu.org/software/octave/about.html [accessed January 19, 2020]

[32] OpenFOAM. https://www.openfoam.com/ or https://openfoam.org/ [accessed January 19, 2020]

[33] OpenJDK. https://openjdk.java.net/ [accessed January 19, 2020]

[34] Apache NetBeans. https://netbeans.apache.org/ [accessed January 19, 2020]

[35] Eclipse Foundation. https://www.eclipse.org/ [accessed January 19, 2020]

[36] Commons Math: The Apache Commons Mathematics Library. https://commons.apache.org/proper/commons-math/ [accessed January 19, 2020]

[37] Hipparchus: a mathematics Library. https://www.hipparchus.org/ [accessed January 19, 2020]

[38] The Modelica Association. https://modelica.org/ [accessed January 19, 2020]

[39] OpenModelica. https://openmodelica.org/ [accessed January 19, 2020]

[40] Jmodelica.org. https://jmodelica.org/ [accessed January 19, 2020]

[41] Gazebo. http://gazebosim.org/ [accessed December 30, 2019]

[42] Koenig, N., and Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2149-2154). IEEE.

[43] The SPICE Page. http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/UserGuide/overview_fr.html [accessed December 28, 2019]

[44] eCircuit Center. About SPICE, http://www.ecircuitcenter.com/AboutSPICE.htm [accessed December 28, 2019]

[45] Hmurcik, L. V., Hettinger, M., Gottschalck, K. S., and Fitchen, F. C. (1990). SPICE applications to an undergraduate electronics program. IEEE Transactions on Education, 33(2), 183-189.

[46] Starzak, L., Swiercz, B., Zubert, M., and Napieralski, A. (2003, February). SPICE-based simulation website: application to teaching of power electronics. In The Experience of Designing

and Application of CAD Systems in Microelectronics, 2003. CADSM 2003. Proceedings of the 7th International Conference. (pp. 334-336). IEEE.

[47] Scilab. https://www.scilab.org/ [accessed January 19, 2020]

[48] Gnuplot homepage. http://www.gnuplot.info/ [accessed January 19, 2020]

[49] Arduino. https://www.arduino.cc/ [accessed January 19, 2020]

[50] Alternative Arduino Interfaces. https://learn.sparkfun.com/tutorials/alternative-arduino-interfaces/all [accessed January 19, 2020]

[51] Raspberrypi.org. https://www.raspberrypi.org/ [accessed January 19, 2020]

[52] Adafruit.com. https://www.adafruit.com/ [accessed January 19, 2020]

[53] RPi Labs. https://rpi.science.uoit.ca/lab/ [accessed January 19, 2020]

[54] Beagleboard.org. https://beagleboard.org/ [accessed January 19, 2020]

[55] ROS.org. http://wiki.ros.org/ROS [accessed December 30, 2019]

[56] The Robotics Back-End. https://roboticsbackend.com/what-is-ros/ [accessed December 30, 2019]