



Robot Racing from Targeted Kit-based Components to a Functional System

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. Michael D. Cook, Milwaukee School of Engineering

Michael D. Cook is an assistant professor in the Mechanical Engineering Department at the Milwaukee School of Engineering (MSOE). He received the B.S. degree in electrical engineering from the University of Wisconsin-Madison, Madison, WI, USA, and the M.S. and Ph.D. degrees in mechanical engineering from Michigan Technological University, Houghton, MI, USA. His interests are in control system design and optimization of mixed-physics dynamic systems, with current research in power flow control with emphasis on the optimization and decentralized control of microgrids.

Dr. William C Farrow, Milwaukee School of Engineering

Dr. WILLIAM C. FARROW has been teaching at the Milwaukee School of Engineering full time for 10 years in the Mechanical Engineering department. Besides teaching courses related to engineering design and engineering mechanics he works with students pursuing aerospace career goals. Dr. Farrow has worked for McDonnell Aircraft Comp., Eaton Corporation's Corporate Research Division, and at NASA's Jet Propulsion Lab as a Faculty Research Fellow.

Robot Racing from Targeted Kit-based Components to a Functional System

Abstract

Affordable computing power and open source hardware have provided many opportunities to enhance STEM education for students. Consequently, numerous electronic retailers offer a diverse array of electronic or educational kits, however without a structured pedagogical framework, students without any prior experience simply learn how to interact with individual components and miss out on how they can be integrated into a system. In this paper, we present our experience in implementing a freshman mechanical engineering course focused on the following main objectives: 1) promoting a computation mindset, 2) providing opportunities to develop essential troubleshooting skills of hardware and software programs, 3) encouraging programming multilingualism and 4) stimulating self-learning and exploration of new hardware to foster lifelong learning skills. The students' lab experiences begin with structured lessons plans regarding analog and digital components and culminates with an open-ended project where students are tasked with developing a robot racer to compete with other students. During the course, students begin programming with MATLAB to reinforce engineering programming concepts and transition to using C programming to implement an embedded solution. A survey was provided to learn about the student's experiences and to help improve future course offerings.

Introduction

Advances in technology and the increase in affordable computational power have enabled the development of more autonomous and dynamic machines with human-like intelligence. The emergence of this technology has brought forth the need to educate highly skilled and computational minded engineers that can solve the complex technical problems of tomorrow to enable the creation of smart machines that can improve our comfort and well-being. For students to be well prepared to take full advantage of the emerging technologies they need to be computationally minded and understand how to process and plan the solutions to difficult and challenging problems by leveraging computational tools. "Computation thinking", as many authors underline, is a fundamental skill that should be part of everyone's analytical toolbelt and is no longer just reserved for programmers or computer scientist [1] and [2]. Computational thinking (CT) as defined by Jeannette Wing, who first brought it to the attention of the computer science education community in 2006 [2], and later refined the definition, "is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer-human or machine-can effectively carry out" [3]. In other words, is a methodology that can be employed to plan and formulate the solution to a problem so that the steps necessary can be carried out by either a computer or a person. One characterization that is used to define the CT involves the following four core cornerstones: 1) decomposition, 2) pattern recognition, 3) abstraction, and 4) algorithms [4]. Decomposition involves breaking the original problem into smaller, more manageable problems that are easier to solve. Pattern recognition on the other hand, deals with being able to identify how similar problems have been solved in the past and use that information to solve the current problem at hand. In contrast, abstraction is a skill used to focus on the relevant information of the problem while ignoring less important details. Lastly, algorithms define the

roadmap or set of steps to be used to solve the problem. Other more detailed characterizations of CT are also used as in [5].

In addition to the wide accessibility of affordable computational power, opensource low-cost hardware have also provided opportunities to more easily integrate system thinking into STEM education. Electronic retailers like Sparkfun, Adafruit, Seeed, and others offer a diverse array of electronic or educational kits which provide students with individual components that include actuators, sensors, embedded computational prototyping boards, and other hardware add-ons or shields. Students don't have to worry about designing the individual components and can focus on using a systems approach thinking for solving more complex and realistic problems. A systems approach thinking, unlike the computation thinking, focuses on how all the individual components or elements as a whole act together to form a system that achieves a common goal or purpose [6], [7], and [8]. It provides students with a wider view that involves a larger number of interactions from the individual components to influence the overall performance of the system behavior. However, system thinking is a skill that is gradually developed through experience and training. From the research literature one way to develop system thinking is by having students work on practical projects that relates to their studies. In turn, the practical projects provide the opportunity for students to learn about the various characteristics of the system by holding distinct roles within a dynamic group environment [8], and [9].

Both computational and system thinking methods provide complementary frameworks that allow students to decompose a complex design problem into more manageable algorithmic based problems. This permits them to envision how to incorporate the various hardware elements into a system with an intended purpose beyond the individual functionality of the various components. However, for students to implement computational and system thinking they must be able to program using a wide selection of available programming languages in order to design algorithms and write programs that can control hardware and solve challenging practical engineering problems.

Two common programming languages taught in many mechanical engineering programs are MATLAB and/or C. MATLAB is a proprietary interpreted programming language that is easier to learn than other languages and offers many numerical methods and a strong visualization environment. C programming on the other hand is a widely available and mature standard language that is commonly used in industry but tends to be more difficult for students to learn. However, C programming has the advantage that it can be used to program embedded solutions using low-cost computation platforms such as Arduino, Raspberry Pi, and Beaglebone. The decision to choose between MATLAB and C programming often creates heated debates in engineering departments as discussed in [10]. However, students who learn to program in C often have an easier time transitioning to other programming languages than those that learn MATLAB because of the similarities to C and C++. Regardless of the primary programming language learned, becoming bilingual or multilingual helps students understand the strengths and weakness of a particular language and helps students better understand their primary programming language [10]. It also creates a clear picture of how programming languages fit together. Students also become more adaptable and can learn a similar programming language paradigm at a faster pace since the

programming skills learned are easily transferable. With the fast-changing computational landscape, it is possible that new programming languages will emerge, and students need to be able to adapt. Lastly, having learned an arsenal of various computer languages expands the students' computational toolbox to be able to solve problems [11].

The main goal of this paper is to present our experience in implementing a freshman mechanical engineering course focused on the following main objectives: 1) promoting a computation mindset, 2) providing opportunities to develop essential troubleshooting skills of hardware and software programs, 3) encouraging programming multilingualism and 4) stimulating self-learning and exploration of new hardware to foster lifelong learning skills. Our approach drew inspiration from computational and system thinking while promoting programming multilingualism to better prepare students for the engineering challenges they will encounter in the 21st century. The class is designed to achieve the following ABET outcomes: *(a) an ability to apply knowledge of mathematics, science, and engineering (b) an ability to design and conduct experiments, as well as to analyze and interpret data (d) an ability to function on multidisciplinary teams (g) an ability to communicate effectively, and (k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practices*

Course Description

The ME1301: Introduction to Mechatronics course is a freshmen-level mechanical engineering class at the Milwaukee School of Engineering (MSOE). The course consists of an 11-week quarter that has a 1-hour lecture meeting twice each week and a 2-hour lab session each week. The purpose of the course is to introduce students to the basic principles of analog and digital I/O and have students apply programming and algorithm development methods to acquire sensor measurements and control hardware. The students gain valuable hands-on experience working with various sensors, actuators, and other hardware. The laboratory topics covered during the class are:


MATLAB Programming

- Week 1: Discrete I/O with switches and LEDs
- Week 2: Analog I/O with DC motors, photocells, temperature sensors, force sensors, infrared sensors, and potentiometers
- Week 3: PWM and servo motor control
- Week 4: Coordinated control of a multi-axis servo motor robotic arm
- Week 5: Introduction to stepper motors and two-axis positioning system
- Week 7: 1D and 2D Scanning with two-axis positioning system

Arduino C-Programming

- Week 6: Introduction to C programming concepts and the Arduino IDE
 - Rewriting week 1 MATLAB Program to C
 - Coordinated control of stepper motor driven two-axis positioning system to:
 - Drawing a rectangle with specified dimensions
 - Implementing Bresenham's line drawing algorithm for 2 Octants
- Week 8-10: Project: Deploying an embedded solution with the Arduino microcontroller with the development of the Robot Racer

The prerequisite for the class is ME 190: Computer Application in Engineering I which introduces students to programming concepts using MATLAB to solve engineering problems. The ME 1301 course is structured to allow students to continue to build on their previous program ability from ME 190 while at the same time introducing them to mechatronic concepts, developing their troubleshooting skills and computational abilities. The students purchase a class development kit from Adafruit as show in Table 1.

Table 1: Class development kit	
ITEM	
	Wheel for Micro Continuous Rotation FS90R Servo PID: 2744
	Continuous Rotation Micro Servo - FS90R PID: 2442
	Adafruit MetroX Classic Kit - Experimentation Kit for Metro 328 PID: 170
	Premium Female/Male 'Extension' Jumper Wires - 20 x 12" PID: 1952
	Orange and Clear TT Motor Wheel for TT DC Gearbox Motor PID: 3766
	DC Gearbox Motor - "TT Motor" - 200RPM - 3 to 6VDC PID: 3777
	5 x AA Battery Holder with 2.1mm DC Jack PID: 3456
	TIP120 Power Darlington Transistors - 3 pack PID: 976

The kit along with a two-axis stepper motor position system and a Lynxmotion articulated robot arm are used for the labs in the course. Every week the labs involve the use of hardware to motivate the review of programming concepts and to help extend the students' programming and computational planning abilities. The students are tasked with both individual and group exercises that they must complete with the hardware. Seeing the hardware move or seeing LEDs light up is a very rewarding experience for students. Consequently, students make more concrete connections between programming structures such as *for loops* and *if-statements* and the physical phenomenon that occurs as a result of executing these instructions. The individual exercises are designed to introduce the students to the hardware and to cover the specific programming objectives for the week. On the other hand, the group exercises are typically done in pairs and are designed to involve more complex programming tasks that build upon the individual exercises. For the group exercises students are asked to plan the algorithm using flowcharts and asked to developed specific functions to help break the complex programming assignment into smaller, more manageable parts. In later labs, students are provided less guidance on how to deconstruct the problem and are expected to develop the functions they need to facilitate the solution to the programing task and how to structure their algorithms based on the knowledge they have acquired from previous lab experiences. As an example, a typical individual exercise would consist of having the student wire a set of LEDs and create code to light up the LEDs in a sequential manner for as many times as the user desires. Whereas a group exercise would consist of wiring various LEDs and switches to simulate the complete cycle of a traffic light and walk signal. Additionally, through the course we have developed tutorials to teach and reinforce the importance of debugging (or identifying and fixing errors) programs. Debugging is an important skill that should be developed and reinforced in all aspects of engineering. In many instances, students feel that if the program executes and runs it must be right. With the debugging tutorials the goal is to teach students a methodology to identify and remove not only careless syntax errors but also how to check and

correct for logical errors that tend to cause the most problems and are harder to find. The expectation is for students to be able to execute a program and in a critical manner be able to check that the results are as expected. This skill of being able to verify that a simulation or analysis is done correctly and provides accurate and credible results is essential and critical in engineering. Through these debugging experiences it is our hope that students get into the habit of looking at their results and ask themselves whether a result is correct, even after the program executes successfully. Through the course, an incremental development strategy for the creation of their algorithms was also encouraged. This involved writing small segments of code and testing them to verify that they work as intended before continuing with further development. The debugging and incremental strategies were not only encouraged for troubleshooting and the creation of algorithms but were also applicable for troubleshooting and constructing intricate circuits.

Another goal of the course was to encourage programming multilingualism. The course begins with students programming in MATLAB to provide a familiar language for students to hone their programming and computational skills. However, a little more than half-way through the quarter, during Week 6 of 10, students are transitioned to programming in C. Nevertheless, this is not the first time they have seen the C programming language. Throughout the course, comparisons between MATLAB and C programs have been discussed with students during lecture. However, this is the first time they must write a C-program on their own. The philosophy is to show students that their programming skills from one computer language to another are transferable. The syntax of the programming languages may be different, but the logic and looping structures function the same way. The idea is to remove students' fear of learning a new language. Once they realize that the logic between languages are the same, they will be more willing to learn a new programming language in the future outside of class. To facilitate the transition from MATLAB to C, a handout with sample code showing the MATLAB and C equivalents is provided to students. During Week 6, in addition to writing their first program in C, students also must follow a prescribed algorithm. The Brensenham's line drawing algorithm is discussed in class without considering the implementation in C and students are asked to understand the algorithm and to implement it in a C-program. This allows students to further hone their computation mindset by observing how a complex procedure is decomposed and planned by another author. This also forces students to have to implement the algorithm exactly as intended and does not leave room for any other interpretation. After concluding Week 6, students take advantage of MATLAB's graphing capabilities and perform a 1D and 2D scanning with a two-axis stepper motor positioning system and IR sensor. The lab requires students to use nested-loops, vectors, 2D arrays and to develop an efficient algorithm to scan a square area with dimensions specified by the user. After Week 7 students are introduced to the project where they will have to develop an embedded solution using the Arduino microcontroller to create a robot racer using their Arduino base kit. The project also allows students the opportunity to learn about new hardware as discussed in the following section.

Final Robot Racing Project

The goals of the project were to have students work in pairs to creatively demonstrate the concepts taught in the course by participating in a high intensity robot race competition. The project requires the team to combine their class development kits, as each kit only comes with one wheel. The design objective was to design, build, program and demonstrate speed and steering control of a robot vehicle, where the robot must be able to complete the race without any physical human intervention. This requires students to take a systems approach to employing the individual I/O

devices covered in the course. Students were also tasked with finding a method to estimate the overall distance travelled with no more than 15% error. Additionally to encourage students to develop and foster lifelong learning skills they were required to use a Liquid Crystal Display (LCD) which functionality and usage was not covered during the course. The LCD was to be used to display the robot commands and to report the estimated overall distance travelled. In addition to the LCD, students could also earn extra-credit points on the project for investigating how to drive the motor forward and in reverse. Achieving this would require the use of an H-bridge or relay. This again provided students with an incentive to research how to do this on their own. The base kit also provided additional components such as an IR sensor and IR remote, mechanical relay, and a piezo buzzer to encourage further exploration and lifelong learning.

The competition consisted of completing a 22-foot-long course in the shortest amount of time and consisted of straightaways, and left and right turns. The students were also given constraints on the physical size of the robot, the distance measurement performance, and budget. They also had to adhere to specific functionality, power, testing, and safety requirements.

Students were also instructed to use best programming practices in the development of their algorithms. These included (1) using an incremental development strategy for the creation of their algorithms, (2) writing and testing small segments of code to verify intended functionality before continuing with further development, (3) using a modular programming approach where the main program is broken down into various user defined functions that perform specific tasks which can be debugged independently of the main program and allows students to work on different aspects of the program at the same time. In addition to the best programming practices previously mentioned, students were required to document all their code with sufficient comments and descriptive variable names.

To encourage the use of the best programming practices students were required to keep track of the functions and aspects of the code that they developed and were graded individually on how well they followed the best programming practices. Students were also evaluated on how well they were able to distribute the project programming tasks.

A detailed description of the project specifications are provided below:

Robot Specifications

1.1 Sensor and Control Requirements

1. Robot must be controlled by an Arduino microcontroller and programmed using Arduino C
2. Input Controls or Sensors should be developed or incorporated to:
 - a. Control the motion of the robot
 - b. Controls should be easy to use to command motion to be able to win a race
 - c. Measure distance travelled with no more than 15% error
 - d. The estimated measured distance and commanded motion should be shown on an LCD screen
3. If the robot does not receive any input control information from the user for more than 45-seconds, it should shut down all functionality and display a message on the LCD display describing the current state

1.2 Power and Propulsion Requirements

1. Robot motors should be powered by 4 AA batteries

2. Arduino microcontroller should be powered by a separate 9V battery
3. Transistors (TIP 120) must be used to control geared DC motors
4. Robot must be capable of driving in a straight-line, make left and right turns
5. Be able to drive forward and in reverse

1.3 Size and Center of Mass Requirement

1. Robot must be able to fit inside an 8"x7"x7" box
2. Center of mass should be kept low to avoid tipping
3. More weight should be distributed above the wheels for better mobility

1.4 Safety Requirements

1. All components must be securely attached to avoid any malfunction during the task
2. No sharp or protruding edges/corners
3. Robot must have a safety kill switch

1.5 Cost Requirements

1. Additional build components not included in the class Arduino kit, should not exceed \$20. The cost of consumables such as batteries are excluded from this requirement.

1.6 Testing Requirements

1. The robot must be able to complete a high intensity 22 feet long race in less than 3 minutes.
2. Robot driver will only be able to control the robot with user inputs and **CANNOT** touch the robot after race starts
3. The robot's ability to estimate distance and display information will be tested
4. Check your robot meets all robot specifications
5. Operation of the robot must be demonstrated and verified by the last day of class of week 10.

1.7 Product Deliverables

1. Working instrumented robot racer by Week 10
2. Written Formal Report (Due on the scheduled final exam date)

Project Schedule

To ensure that the students made significant progress each week the following project milestones were specified for weeks 8-10.

- **Week 8:** The robot must be built. Must show motors moving and LCD displaying messages.
- **Week 9:** The robot must demonstrate variable speed forward and left/right turning. It should also be able to display the distance travelled. Inputs to control robot should be incorporated.
- **Week 10:** The robot must be able to compete in race on the 2-hour lab day and meet all robot specifications. Project must be fully functional.

Student Projects

Some examples of team projects are described in the proceeding section.

Project 1:

A team of two students worked on this project. Figures 1 and 2 show the overall layout of the robot racer and the IR remote used. The design incorporates two DC motors with wheels, motor driver, LCD screen, Arduino microcontroller, custom chassis, caster wheel, batteries, potentiometer, IR sensor and remote control. The motor driver is used to control the speed of the wheels while any commands sent to the robot are displayed on the LCD screen. The brightness of the LCD screen is controlled by adjusting the potentiometer. The IR pulses transmitted from the IR remote are received by the IR sensor and are decoded in software to interpret which button was pressed. The Arduino is powered using a 9-volt battery. Due to the internal resistance of common 9-volt batteries and the current drawn by the motors, a separate 9-V power pack was employed to power the DC motors to prevent (1) underpowering the Arduino during times of high current demand (2) overheating the Arduino's on-board voltage regulator, and (3) accidentally violating any of the Arduino's individual pins max current rating.

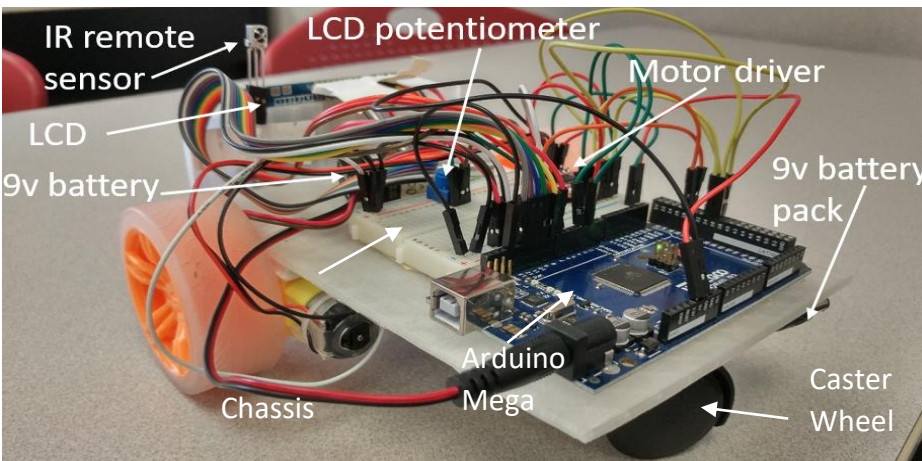


Figure 1: Robot layout of components

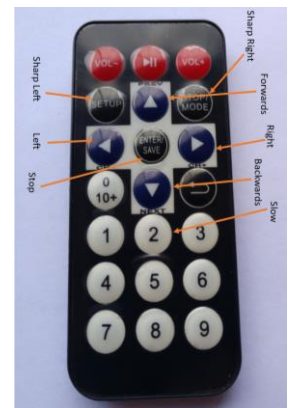


Figure 2: IR remote used

Figure 3 shows the wiring diagrams created by the team to document how the DC motors are powered and how the motor controller is interfaced with the Arduino microcontroller. As shown in Figure 3, the logic of the motor controller is powered directly from the Arduino 5-V rail while the motors are powered from a separate 9-V supply.

Figure 4 illustrates the required connections needed to power the LCD from the Arduino, the necessary wiring to incorporate a potentiometer to adjust the brightness of the screen, and the digital I/O's used to write to the screen. Both Figure 3 and 4 were created using Fritzing, a free to build open-source software program.

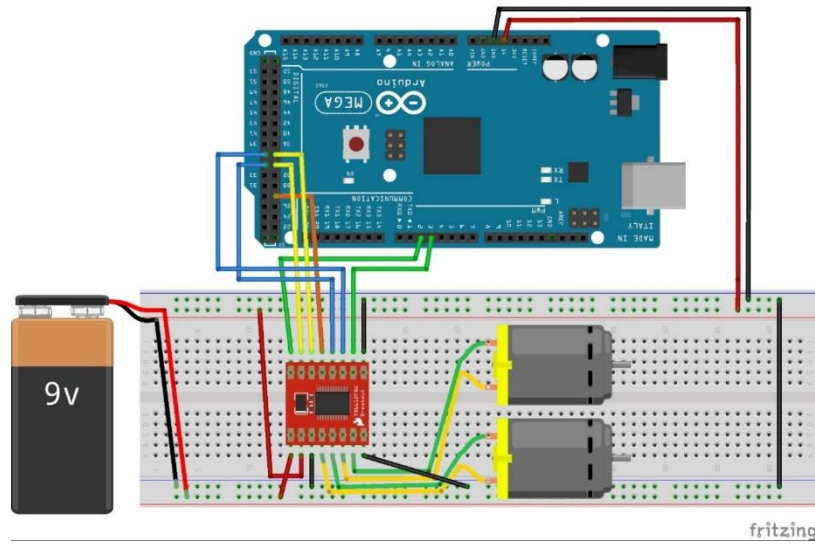


Figure 3: Wiring Schematic for DC Motors and motor driver. Diagram created with Fritzing [12]

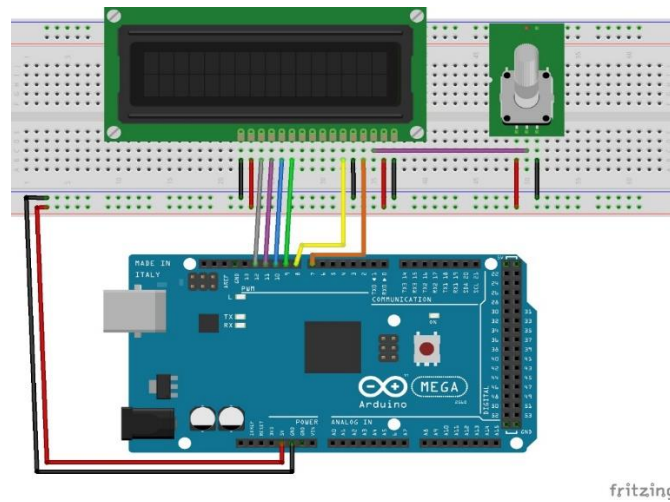


Figure 4: Wiring Schematic for the Liquid Crystal Display and Contrast Potentiometer. Diagram created with Fritzing [12]

Some unique self-learning characteristics of this project, in addition to the required integration of the LCD, included the use of an infrared remote control to command the robot. Students had to learn how the IR sensor and remote kit worked and explore the Arduino IR library to interpret the IR signals. Additionally, students incorporated a motor driver which was not introduced in the course to drive the motors both forward and in reverse. To document the project circuits, students had the choice to use simple PowerPoint drawings but instead opted to learn how to use the Fritzing program on their own to create more realistic and visually impacting diagrams. Overall, approximately 81% of the class used the Fritzing program to document their circuits.

To determine the overall distance travelled by the robot, the team first performed an experiment to find the time required to travel 3 meters at different speed settings. From this information the speed of the robot was determined for the three speed settings and used to estimate the distance.

The distance was calculated by computing the distance travelled every tenth of a second and adding this distance to the total travelled distance. One of the issues that the team encountered was that their distance calculation measurement was sensitive to the battery voltage and required other speed calibrations as the voltage changed due to usage.

Project 2:

A second team consisting of two students also used an IR remote control kit to control the robot motion. However, their approach to controlling the direction of the DC motor rotation was different. Instead of using a motor controller like the students from Project 1, the students used transistors and mechanical relays to control the direction of current flow and consequently the direction of the motor rotation. Students during the course were taught how to drive a motor in one direction, however no methods were presented to reverse the polarity. Armed with the necessary foundational knowledge from the course, the team researched, learned, and understood how to reverse the motor polarity to drive the motor in both directions. The circuit for the drive train of the robot can be seen in the Fritzing diagram in Figure 5.

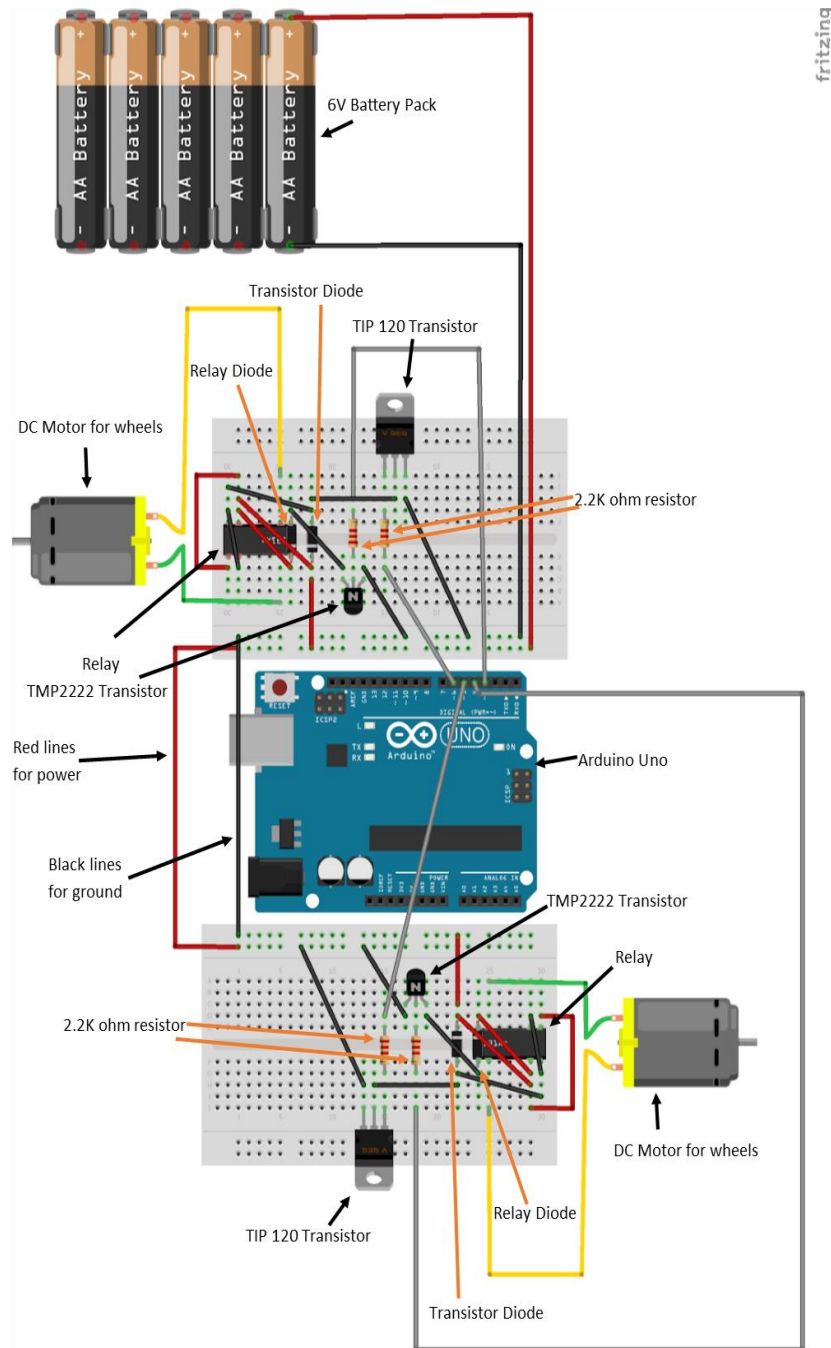


Figure 5: Robot drive train circuit. Diagram created with Fritzing [12]

Texas Instruments Power (TIP) 120 transistors were used to drive the motors while the P2N2222AG transistors were used to energize the coil in the relays.

Another interesting aspect of this team's project was the method they used to measure the distance travelled. While many teams used a speed calibration approach as described in Project 1, this team developed their own custom optical encoder using the robot's wheel, electrical tape, and a light sensor, as shown in Figure 6.



Figure 6: Custom encoder developed by the team to measure position

Covering part of the wheel with electrical tape created a small window that allowed light through for part of the wheel rotational cycle while the rest of the time the inside of the wheel was dark. This variation between high and low light intensities inside the wheel was detected by the light sensor and was used to count the number of rotations of the wheel. From the number of rotations, the team was able to determine the overall distance travelled by multiplying it with the circumference of the wheel. To ensure good encoder performance the team obtained the dark and light intensity values inside the wheel from the sensor to determine an appropriate light threshold for determining the rotation of the wheel. This reduced the false positives that could result from the light variations in the room.

Project 3

Another team consisting of two students took a different approach for controlling the robot motion. Instead of using an IR sensor and remote they developed a wired solution by using a secondary breadboard, potentiometers and switches, as shown in Figure 7. The two potentiometers were used to control the steering angle of the robot and the motor velocity. The switches were used to implement an emergency kill switch and an ignition button.

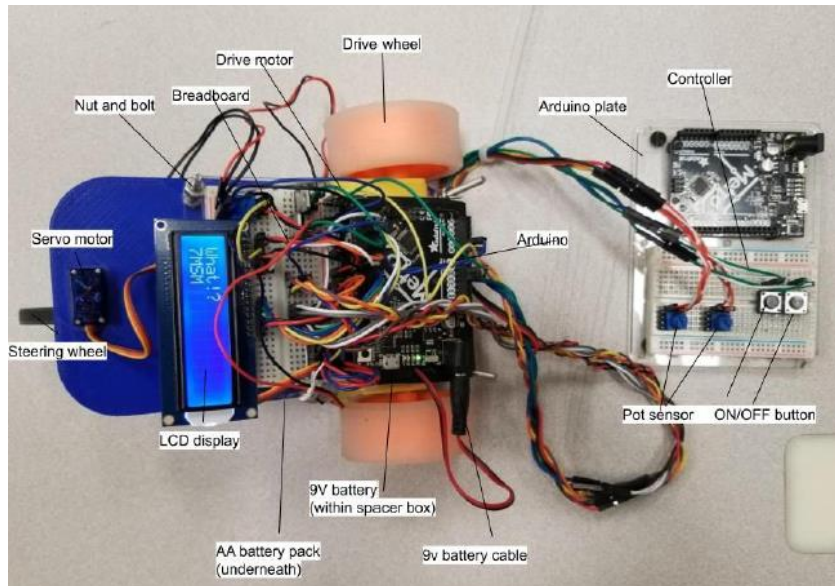


Figure 7: Robot layout of components

This team utilized a micro servo motor to steer the robot, an unexpected student solution to steering; all other teams used a differential drive approach to steer the robot vehicle. The steering mechanism allowed the team greater control of the robot's direction. A side view of the vehicle illustrating the steering mechanism can be seen in Figure 8. To manufacture the steering mechanism and the chassis of the robot, the team 3D printed the components.

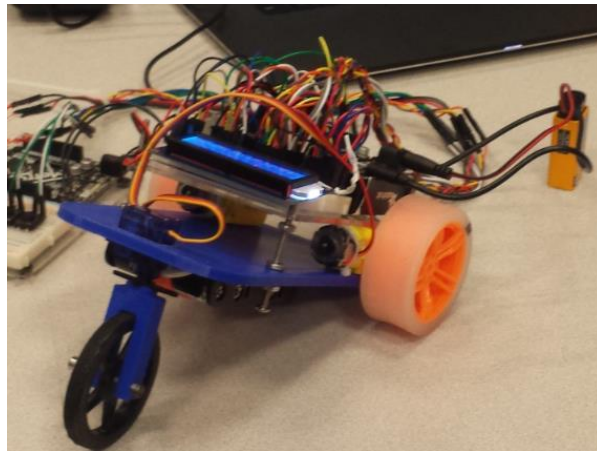


Figure 8: Steering robot mechanism manufactured by the team

Conclusion

This paper describes the structured pedagogical framework used in a freshman mechanical engineering course that uses a targeted class kit to teach students how to take individual components and transform them into an integrated system. The course focused on

promoting a computation mindset by having students deconstruct a complex task into more manageable and modular tasks, and by requiring students to plan their algorithms using flowcharts. Furthermore, a system thinking approach was encouraged with the project, as it required students to take the individual components formally covered in the course and combine them into an integrated system that met specific design requirements. The course also encouraged programming multilingualism by introducing students to a new programming language and drawing parallels between the two languages to mitigate the fears of learning a new language in the future. The class also provided opportunities for students to develop essential software and hardware troubleshooting skills with the use of tutorials and with the creation of weekly circuits. Furthermore, the class culminated with an open-ended project where students were tasked with developing a robot racer to compete with other students. This project was designed to promote and stimulate self-learning opportunities and to provide students with an intricate task that required the students to break the problem into smaller tractable subproblems, examine feasible solutions, and develop a winning algorithm to beat the competition. A student survey was conducted at the conclusion of the course to assess the students' experience with the goal of improving future lab activities. From the survey it was clear that students felt that the project helped them feel more comfortable programming in C. In addition, they appreciated learning C-programming to be able to deploy a stand-alone mechatronics system that did not require a bulky laptop or desktop. Students also reported that the project helped them improve their troubleshooting and programming debugging skills. Finally, students felt confident in learning how to use new hardware that involved digital and analog concepts formally taught in class. Overall the structure of the class and the project was a success. The hands-on lab experiences helped students understand mechatronic concepts, and the project helped create a competitive environment that motivated students to learn how to use new hardware while at the same time creating a level of camaraderie among teams that made the experience fun and enjoyable. In the future, to improve the course, more time will be dedicated to making a transition from MATLAB to C-programming and more activities that help develop student's algorithm abilities will be developed. Finally, the authors would like to emphasize that MATLAB was the primary programming software used for this freshman experience due to its availability at our institution. However, the same experience could have also been accomplished with other suitable open-source software such as GNU Octave, since it provides a similar interface to the Arduino support package in MATLAB. The availability of this open-source software makes this experience more accessible to other institutions with limited resources.

References

- [1] P. J. Denning and M. Tedre, *Computational Thinking*, MITP, 2019.
- [2] J. M. Wing, "Computational Thinking," *Communications of the ACM*, vol. 49, no. 3, p. 33–35, 2006.
- [3] J. Wing, "Computational Thinking Benefits Society," 40th Anniversary Blog of Social Issues in Computing., 10 Jan 2014. [Online]. Available: <http://socialissues.cs.toronto.edu/index.html%3Fp=279.html>. [Accessed 19 March 2020].

- [4] "Introduction to computational thinking," BBC Bitesize, [Online]. Available: <https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1>. [Accessed 19 March 2020].
- [5] S. a. P. R. Grover, "Computational Thinking in K-12: A Review of the State of the Field," *Educational Researcher*, vol. 42, no. 1, pp. 38-43, 2013.
- [6] P. I. William, "Systems," in *System Dynamics*, McGraw-Hill, 2014, pp. 2-3.
- [7] D. Aronson, "Overview of system thinking.," 1996. [Online]. Available: http://www.thinking.net/Systems_Thinking/OverviewSTarticle.pdf. [Accessed 19 March 2020].
- [8] S. K. R. G. a. F. M. Kordova, "Developing systems thinking among engineers: Recent study findings," in *IEEE Systems Conference (SysCon) Proceedings*, Vancouver, BC., 2015.
- [9] S. a. F. M. Kordova, "The T Shape dilemma (depth versus width) in education of industrial engineering & management and its reflection in the students team project," Technion-Israel Institute of Technology, 2010.
- [10] H. H. Cheng, "C for the Course," *ASME Mechanical Engineering Magazine*, pp. 50-52, September 2009.
- [11] M. Kamaruzzaman, "5 reasons to learn a new Programming Language in 2020: Learn a new programming language to boost your career and skillset in the new year," 27 Dec. 2019. [Online]. Available: <https://towardsdatascience.com/5-reasons-to-learn-a-new-programming-language-in-2020-bfc9a4b9a763>. [Accessed 19 March 2020].
- [12] F.-o.-F. Foundation, "Fritzing Electronics Made Easy," [Online]. Available: www.frtizing.org. [Accessed 2018].