Infusing a Series of Motor Lab Exercises and Projects into the Freshman Circuits Course

Abstract

The "Introduction to Electrical Engineering Technology," commonly referred to as the Freshman Circuits Course, is a comprehensive program encompassing three modules—Electrical Circuits, Microcontroller Applications, and Digital Circuits. Developed ten years ago, the original function of the 4.5-hour lecture-lab course presented in two sessions per week, each one for 2hourse and 15 minutes long, was to introduce freshman students to the subject of EET by focusing on the early portions of the sophomore year DC circuits course at a relaxed pace. Another purpose was to have freshmen connect with EET faculty before their sophomore year. In recent semesters, we've seen greater numbers of freshmen lacking general tooling experience and being unprepared in basic math, and in response, we've added more hands-on exercises and projects and included more practical algebra-based word problems. This past year, we incorporated a servo motor lab sequence in one of the course sections and a DC motor sequence in the other section.

This paper focuses on a plan to infuse a broad series of motor-based labs and projects into the Freshman Circuits Course, aiming to increase interest and motivation by demonstrating that some very understandable and easily modified Arduino code can control different aspects of motor action (Arduino software application was developed with coding simplicity in mind). We hope that adding these lab exercises, where students can modify the code and view corresponding changes in motor operation, will inspire students to progress through our program. And help bridge the gap between theoretical knowledge and real-world applications. The motors selected for exploration include DC motors, servo motors, and the planned integration of stepper motors. The paper outlines the course design, specific lectures, labs for each motor type, and course evaluation methodologies and concludes with insights drawn from the ongoing improvement efforts.

1. Introduction

The course, also known as the " Freshman Circuits Course," is structured to provide Electrical Engineering Technology students with a preparatory understanding of electrical circuits, digital circuits, and microcontroller applications. This paper outlines the design and integration of motor-based labs and projects to enhance student engagement and application-oriented learning.

Infusing motor applications could address the perceived lack of technical experience among freshmen. The EET program at Buffalo State originated as a 2+2 program where incoming students were juniors with a 2-year technical degree and often worked in the field as an associate degree technician. For the past two decades, the program has offered a complete 4-year set of courses, and incoming freshmen are younger, lack technical backgrounds, and are more unprepared in mathematics [1].

Students in the DC Motor section seemed excited to test out their code for moving the robot in a square pattern. They moved back and forth from the delineated square pattern in the hallway outside the lab to their laptop and robot holder in the lab. Two of the ten students chose to take the DC motor option for their final project. Meanwhile, those working on the Servo Motor project had the opportunity to control a garage door, engage in a more straightforward project, and witness firsthand the application of this device in robotics projects. One sample of a Servo motor application in the student's robotic projects has been displayed in this article [2].

Motor modules have been designed and will continue to be developed to address challenges in motivating and retaining incoming first-year students. Integrating new motor exercises provides real-world applications and immerses students in algebra applications within motor parameter problem-solving scenarios. Through ongoing improvements based on student performance and course evaluations, this paper highlights the evolution of the Freshman Circuits Course to create a dynamic and effective learning environment.

2. Course Specifications

This section discusses the course's various aspects, including the textbook's history and the previous and current construction of the course. It also discusses the grading criteria. In section 8, the student's grades are displayed and analyzed.

2.1.Background

The Buffalo State EET program was created in the early 1970s to provide the local electronics industry with applied talent in electrical engineering. The original program had a 2+2 structure, where students who completed their associate degree in electrical technology could enter our program while often working as a 2-year degree technician. The Buffalo State program has offered the complete 4-year structure for several decades. The electrical engineering Technology program includes two majors: Electronics and Smart Grids. The first two years of these have the same courses. Early in the first year of the 4-year structure, freshmen in both majors only took courses outside the major, completing their math, science, and general education requirements. A screenshot of the course sequence for the first year in both Electrical Engineering technology programs is shown in Fig. 1.

The freshman circuits course was developed about ten years ago to address the concern that we did not have a connection with these students until their sophomore year. The original goal of the freshman course was to teach students the early portions of the sophomore year DC circuits course at a relaxed pace. The course has evolved over the recent years to respond to students' lack of informal technical and tooling backgrounds by adding more project-based lab experiences, including activities in the microcontroller and digital circuits areas. Students who entered the program with such backgrounds, as was the case for the 2+2 structure, seemed to be

already motivated by their previous technical experiences. We have also supplemented the course with practical algebra problems to address a lack of math preparedness.

Year; Semester	Course Department Number, Title	Credit Hours				
		Communications	Math and Science	Technical Content	Social Sciences, Arts, and Humanities	Other
1/Fall	CWP 101, College Writing I	3				
	MAT 126 (or MAT 161 + 163) Calculus I		4(5)			
	General Education				3	
	ENT 104, Introduction of EET			3		
	General Education				3	
1/Spring	CWP 102, College Writing II	3				
	MAT 127 (or MAT 162+164) Calculus II		4(5)			
	PHY 107 (or PHY 111) Physics I		4(5)			
	General Education				3	
	General Education				3	
2/Fall	ENT 330 Electric Circuits Analysis I			3		
	PHY 108 (or PHY 112) Physics II		4(5)			
	ENT 300			3		

Fig 1. Recommended Sequence of Courses Electrical Engineering Technology Majors (Electronics/Smart Grid)

2.2. Course Design

The course begins with an introductory module covering essential components and mathematical modeling. It emphasizes Ohm's Law, the application of simulation, and hands-on experiences with breadboard connections to deepen students' comprehension. This groundwork paves the way for exploring the upcoming Motor-based Labs.

Moving to the second module, students work with the Arduino board as an example of a microcontroller system. Applications for small electronics projects involving LEDs, switches, and potentiometers build fundamental Arduino programming skills. After working with basic digital input/output for LEDs and switches, students now explore how Arduino can control different electronic components such as LCDs, Servo Motors, and DC Motors. Other topics, including the significance of pulse width modulation (PWM) in regulating DC motor speed, are discussed and tested in the theory sections. The TinkerCAD simulator is also introduced as an Arduino simulator, which enhances the hands-on approach, providing students with practical insights into software-circuit interactions and fostering project-based learning [3].

In the third module, students delve into the basics of digital logic, covering logic gates, Boolean Algebra, and truth tables. Plans include integrating stepper motor labs to demonstrate digital circuits' control over these components.

The course grades are weighted evenly between analytic work (exams/quizzes) and hands-on skills (individual labs and projects) to highlight the importance of hands-on aspects. To bolster the analytical portion, a junior/senior level student was hired as a dedicated tutor for the freshman circuits course (there were two sections this past semester with approximately 30

students total). In Fig. 2, the learning activities and grading criteria for the Fall 2023 semester have been displayed. The first four chapters of Electronics Fundamentals: Circuits, Devices & Applications, by Thomas L. Floyd, have been used as references. However, students are not required to provide the textbook [4]. All course materials and presentation slides were posted on the course webpage after the lecture, and the solutions to problems in the class were also posted.

Category	% Of Grade	Grade Items (Learning Activities)
Circuit Labs and	20%	➢ Related tasks (Experimental, Report, Teamwork, Simulation,…)
Small Project	15%	➢ Related tasks (Experimental, Report, Teamwork, Simulation,…)
Homework/ Quizzes	20%	➢ A few announced or unannounced will be assigned during the semester with equal weight.
Midterm Exams	30%	➢ Two Exams that each will cover part of the course topics discussed during that period of the semester up to the exam date ➢ The Exam materials are not cumulative, and each one covers only the new materials taught after the last exam
Final Exam	15%	➢ **It is scheduled for 12/11/2023, 1:40-3:30 am.** ➢ The final Exam covers part of the course topics discussed during the last period of the semester. ➢ The Final Exam is not comprehensive, and it covers 80% of the new materials taught after the second midterm and 20% of the materials from the earlier course topics.
TOTAL	**100%**	

Fig 2. Learning Activities and Grading Criteria

3. A Sample Lecture and Lab Taught for the application of DC Motor Control

One of the Freshman Circuit sections added a set of lab exercises involving DC Motor Control to the microcontroller module this past semester. We wanted to provide a real-world example of how a microcontroller system like the Arduino can control a motor. The lab exercises in this

module included consideration of motor current requirements, use of the Arduino time delay function to provide Pulse Width Modulation, and experience following wiring tables. By completing this lab, students prepare to address contemporary challenges in advanced electrical engineering and power systems projects during their senior design project or while working in the industry [5] –[6]. One of our goals was to introduce the concept of a "Transistor Driver," which would allow the 'small' digital control signals from the Arduino to control and drive a DC Motor (see Fig. 3). We also wanted to Introduce Pulse Width Modulation for DC Motor control using Arduino's general-purpose digital IO and the Timing function. We used algebra to design PWM pulse times for a given speed.

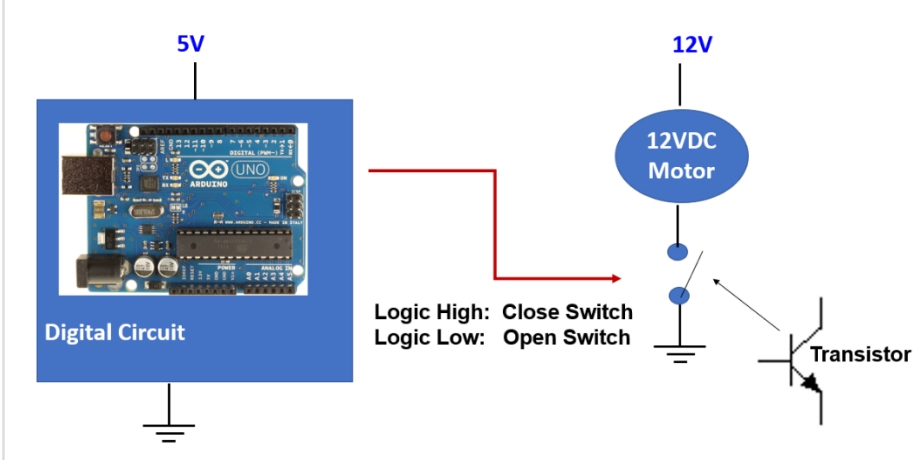


Fig 3. Transistor Driver for Motor

The first DC Motor exercise involved the Arduino's ON/OFF control of a single 12V DC motor, incorporating a breadboarded power transistor driver as an interface. The design technique for selecting the driver was discussed, using Ohm's Law to determine motor current based on the motor coil resistance and transistor current gain. The current limits of the Arduino digital IO lines were compared with the current required by the motor. The transistor driver circuit was initially simulated in Multisim with a resistor to model the motor coil resistance. One of the hopes is to introduce the transistor to first-year students in its most basic application of current gain.

Students were given the code below (Fig. 4), which turns the motor on for 5 seconds (using the Arduino function "delay" and off for 1 second.

The concept of PWM was explored, and together, the class modified their code to provide 80% PWM (8 msec on, 2 msec off) and observe the motor's speed. Time was spent discussing the generally appropriate frequency range for a motor PWM input, converting that to periods, and using the Arduino delay function to achieve the desired pulse width (see Fig. 4).

```
int Base = 3    ; Connection to base resistor

// the setup function runs once when you press reset or power the board
void setup()  {
  pinMode(Base, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(Base, HIGH);   // turn Motor On (BJT switch closes)
  delay(5000);                      // wait for 5 seconds
  digitalWrite(Base, LOW);    // turn Motor off (BJT switch opens)
  delay(1000);                      // wait for a second
}
```

```
int Base = 3    ; Connection to base resistor

// the setup function runs once when you press reset or power the board
void setup()  {
  pinMode(Base, OUTPUT);
}

// the loop function runs over and over again forever
// Motor stays on at 80% PWM
void loop() {
  digitalWrite(Base, HIGH);   // turn Motor On (BJT switch closes)
  delay(8);                       // wait for 8 msec
  digitalWrite(Base, LOW);    // turn Motor off (BJT switch opens)
  delay(2);                       // wait for 2 msec
}
```

Fig 4. Turn the Motor On and OFF and Drive the Motor with PWM 80% of the Supply

In the second exercise, student groups were given a small robotic base controlled by 6 of the same motors. The robotic base included an H-Bridge module for driving two motors. The H-Bridge unit was introduced along with the wiring diagram required for the Arduino to control the robotic base (Fig. 5). The robotic base sat on a wooden block so the wheels could spin while the robotic base remained in place. The initial wiring used the 2 H-Bridge motor controllers to control the three suitable side motors and three left side motors. Students could then experiment with driving the motor forward, backward, and turn. Once the robotic base was up and running, one of the experiments we did was to have the motors travel a fixed distance; students measured the time of the excursion and calculated the motor speed. Students were then tasked with designing code to move the robot in a rectangular path.

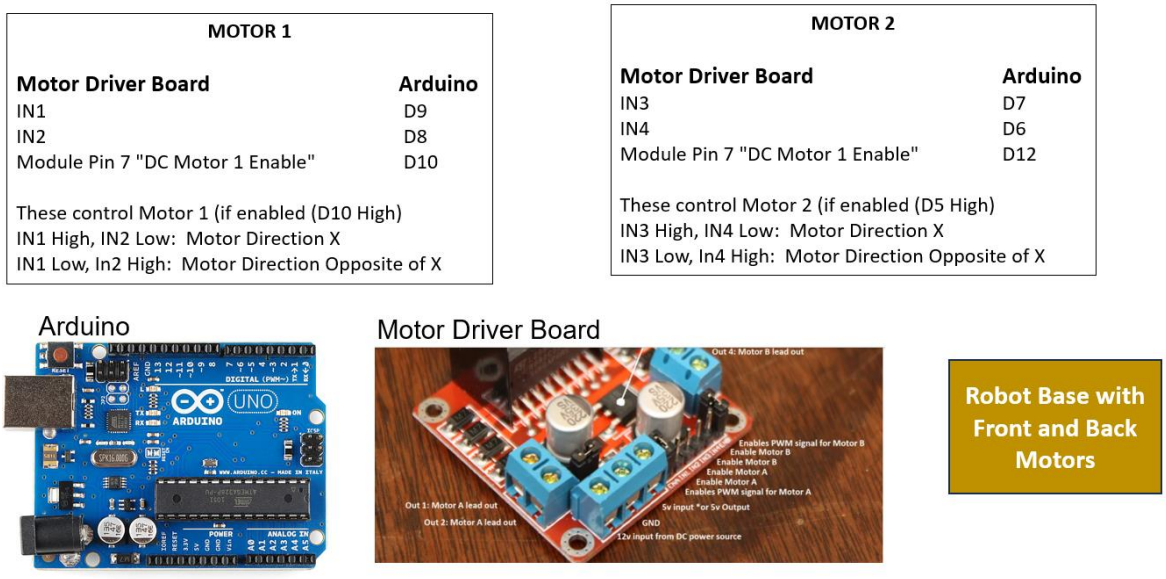


Fig 5. Wiring Information

4. A Sample Lecture and Lab Taught for the application of Servo Motors

The theory for explaining the Servo Motor was presented at the beginning of the class. A shorter version of this explanation has been added to the lab instruction's background to help students access the formula during the experiment.

4.1. The Technology and Application of Servo Motors

Transitioning to servo motors, these specialized positioning motors move to specific positions and stay there until instructed otherwise, making them ideal for controlled applications such as retracting landing gear on model airplanes or adjusting robotic arms. Servos typically rotate 180 degrees and operate based on pulses received at consistent intervals, with pulse width varying between 1000 and 2000 microseconds. The Arduino UNO, with six dedicated PWM pins, specifies the role of PWM pins in servo control for better clarity. The Arduino community further enhances functionality through software libraries, extending the Arduino's capabilities. This collaborative environment fosters contributions, leading to the development of libraries for various sensors and actuators, including those designed for servo motors.

In the following, a practical project called garage door opener was done in the class, where students created a garage door opener using Arduino. The system included a pushbutton, two LEDs (Red and Green), and a servo motor. The Red LED is initially turned on. When the button is pressed, the servo motor rotates to simulate lifting the garage door, and the LEDs switch from Red to Green. After a short pause, the door closes, and the LEDs switch back to their original state. This hands-on project involved indicating assistance availability using a little cardboard craft. The lab instruction comprised multiple sections: background, project setup, programming, project running, simulation, and lab report creation. A summary of the lab instruction has been included. For the coding part, some of the functions and their application were taught earlier in

the class. The code was broken into several parts to organize the coding process, and each module was described individually.

4.2. Project Instruction

Background

Servo motors are specialized for precise positioning, rotating up to 180 degrees. When integrated into a cardboard craft, they offer a signaling mechanism for project assistance. Like PWM for LEDs, servo motors use pulses to indicate rotation angles. Arduino simplifies this with a built-in library for easy motor control. The 'map()' function adjusts values from the potentiometer, accommodating the servo's limited 180-degree rotation and the analog input range of 0-1023. The Arduino community extends functionality through software libraries, including one dedicated to servo motors, enhancing code simplicity and versatility.

Hardware Setup

Referencing the schematic model, build your circuit (see Fig.6).

1. Connect 5V and ground from the Arduino to one side of your breadboard.
2. Place a potentiometer on the breadboard and connect one side to 5V, the other to the ground, and the middle pin to analog pin 0 (A0) for servo control.
3. The servo has three wires: power (red), ground (black), and control (white). Use three male headers to connect them to the breadboard, with 5V to the red wire, ground to the black wire, and the white wire to pin 9.
4. To stabilize voltage during servo movement, add a 100uf capacitor near the servo headers and, optionally, another one across the potentiometer power and ground.
5. Add header pins to connect the servo's female connectors to the breadboard.

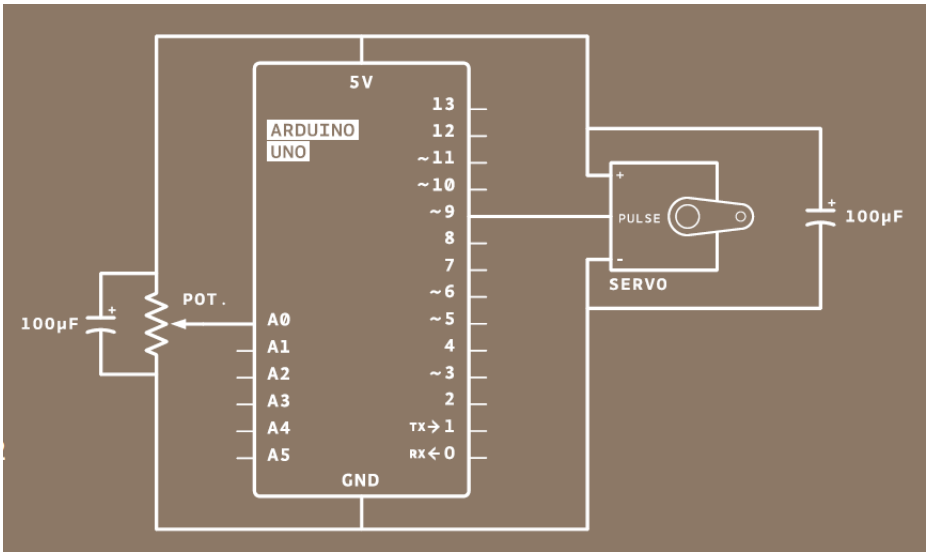


Fig 6. Schematic Model of Servo Project

Programming

1. Import the servo library to make its functions available for your sketch.
2. Create a named instance of the servo library by defining a variable. This variable serves as a unique name with all the capabilities of the servo library, such as "myServo."
3. Set up a constant for the potentiometer pin and declare variables to store the analog input value and the desired servo angle.
4. In the setup(), specify the pin to which your servo is attached.
5. Establish a serial connection to monitor potentiometer values and their corresponding servo angles.
6. Read the analog input in the loop() and print the value to the serial monitor.
7. Use the map() function to scale the analog input values (potVal) from 0 to 1023 to servo angles ranging from 0 to 179.
8. Print the mapped value to the serial monitor.
9. Move the servo using the servo.write() command to the specified angle.
10. Introduce a delay at the end of the loop() to allow the servo time to reach its new position.

Running Project

1. Open the serial monitor after programming your Arduino.
2. Check for continuous values like "potVal : 1023, angle : 179" indicating the potentiometer and servo values.
3. Turn the potentiometer to observe changes in values and the servo's new position.
4. Utilize the full range of potentiometer values (0-1023) for testing projects with analog input.
5. Tape a cardboard arrow to the servo at 90 degrees, aligning it with the motor's body.
6. Draw a half circle on a larger piece of paper, with "Stay Out" on one end, "Come in" on the other, and "Knock please!" in the middle.
7. Attach the servo with the arrow to the paper, creating a visual indicator of your project availability.

Simulation

1. Create a free account on TinkerCAD, an open-source software.
2. Follow the instructions in the uploaded video, How to Design a Circuit in TinkerCAD. It would be best if you made the same circuit you made on the Breadboard.
3. Insert your code from Arduino Uno into the code area of the simulator.
4. Run the simulation and change the potentiometer status.
5. Take a clear screenshot of your circuit, the code, and the Serial Monitor display.

Deliverable:

1. A short video of running the circuit shows how you can control the output of the Servo motor using the potentiometer to point to the three positions on paper ("Stay Out," "Come in," and "Knock please!").
2. A screenshot of your simulation.
3. Your program .ino file has to be uploaded on Brightspace.
    a. You need to add a title box including your names and the date at the beginning of your program.
    b. Also, you need to comment on your code. Some points will be deducted for missing these parts.
4. Summarize key learnings and observations from the servo motor integration project, emphasizing the practical application of Arduino in creating a signaling system for project assistance. Compare running your project in the simulator and the actual circuit.
5. The rubric in the assignment will evaluate your submissions (see Fig. 7).

104 F23 Project1 Rubric

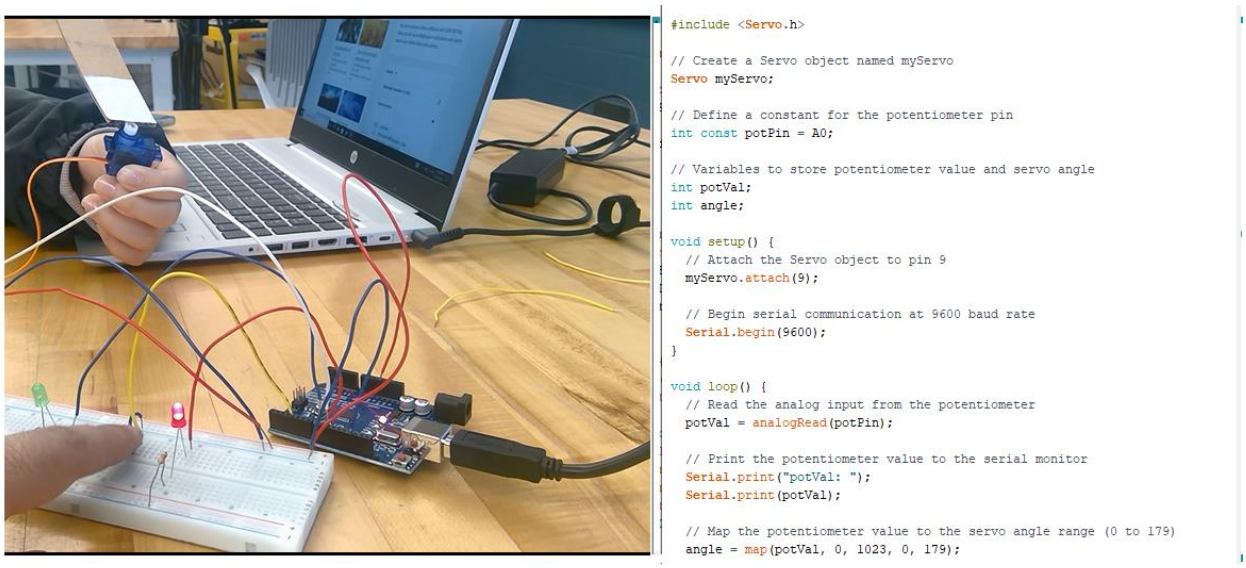
Course: 23FA ESSENTIALS OF EET 3393

Criteria	Level 4	Level 3	Level 2	Level 1	Criterion Score
Code	25 points	20 points	15 points	0 points	/ 25
	The code is correct, efficient, and working. It is well written, commented and has a title block.	The code is correct, not efficient, but working There are some comments	The code is partially correct	The code is wrong. The file is not included or can't be opened.	
Video	25 points		15 points	0 points	/ 25
	The video is running and clearly shows the setup.		The video is running, but the setup is not shown	No video is included, or the file cannot be opened.	
Screenshot of TinkerCAD Circuit	25 points		15 points	0 points	/ 25
	The screenshot clearly shows the setup, code, and Serial Monitor display while running the circuit.		Some parts are missing, but the circuit still runs without error.	No picture is included, or the file can not be opened.	
Conclusion	25 points		15 points	0 points	/25
	One paragraph of the result's analysis is needed. A comparison is made, and the questions are answered.			No correct answers	

Fig 7. Rubric for evaluation of the lab reports

4.3.Sample Students Report

Each group has submitted a video of running the project along with code and analysis. Figures 8 and 9 show a screenshot of the student's sample report for experimental and simulation results.



```
#include <Servo.h>

// Create a Servo object named myServo
Servo myServo;

// Define a constant for the potentiometer pin
int const potPin = A0;

// Variables to store potentiometer value and servo angle
int potVal;
int angle;

void setup() {
  // Attach the Servo object to pin 9
  myServo.attach(9);

  // Begin serial communication at 9600 baud rate
  Serial.begin(9600);
}

void loop() {
  // Read the analog input from the potentiometer
  potVal = analogRead(potPin);

  // Print the potentiometer value to the serial monitor
  Serial.print("potVal: ");
  Serial.print(potVal);

  // Map the potentiometer value to the servo angle range (0 to 179)
  angle = map(potVal, 0, 1023, 0, 179);
```

Fig 8. A screenshot of the video showing the working setup

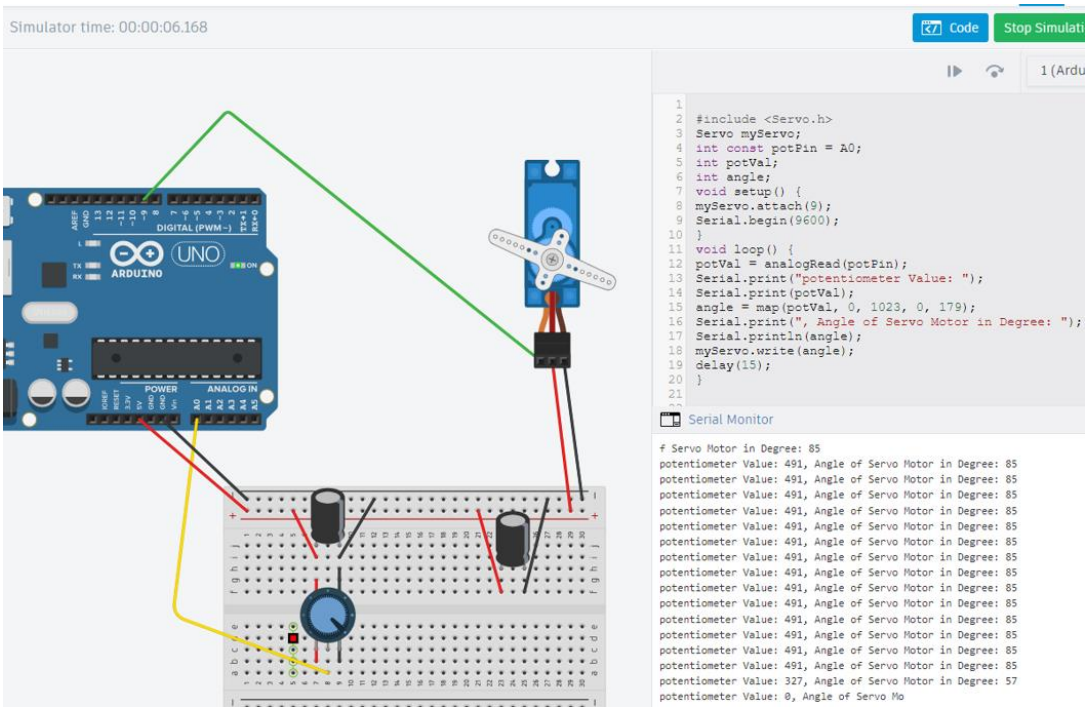


Fig 9. A screenshot of the TinkerCAD simulation

5. A Sample Lecture and Lab is designed for the application of Stepper Motor Control

A Stepper Motor Control module is being designed for the upcoming fall semester Freshmen Circuits course (Fall 2024). For simplicity, we chose a Unipolar Stepper Motor so that a similar

Transistor Driver circuit can control each phase. Initial experiments will use a flip-flop shift register circuit to control the motor, providing an example of a Sequential digital circuit. Currently, the Digital Circuits module only contains Combinational circuits consisting of fundamental AND, OR, and NOT gates. The shift register will be breadboarded and tested with our Digital Trainers. Students will provide a 'Manual Clock Signal' to step the motor and use the Digital Trainer clock signal for stepping. We will replace the shift register with an Arduino to control the stepper motor. Students will use algebra to calculate 'Step Timing' to obtain a given motor speed for the specific step size of the motor. We will be adding some positioning exercises as well. We will add positioning exercises to position the stepper motor relative to a user-controlled potentiometer (see Fig. 10)

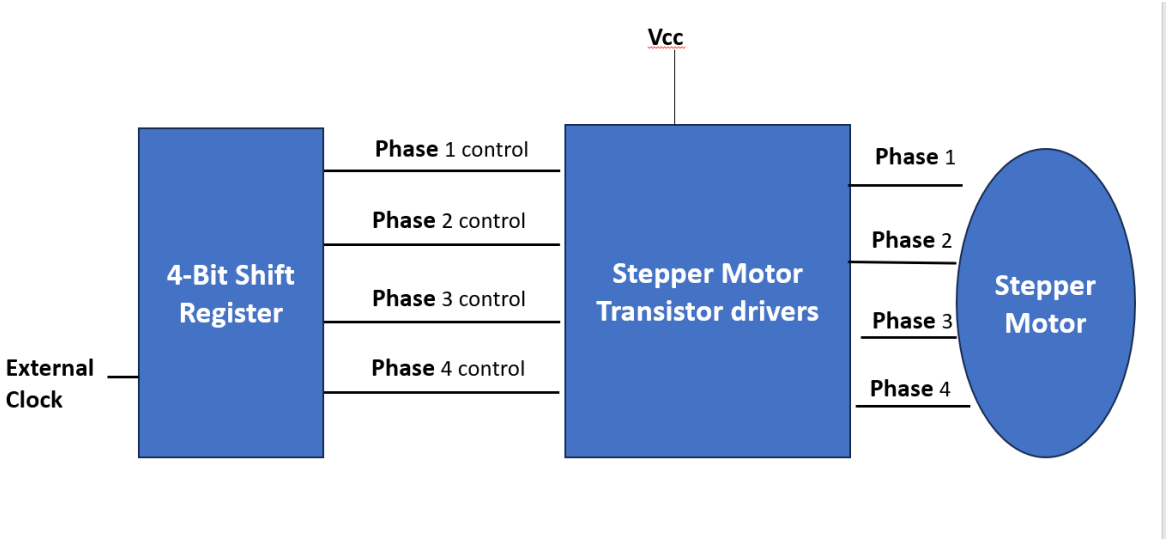


Fig 10. Example for 4-Phase Unipolar Stepper

6. Analysis of Arduino Projects and Student Academic Performance

The course schedule in Fig.1 displays learning activities, including circuit-based lab experiments and Arduino projects, defined for evaluating the student's performance in this course.

Table 1. Semester-wise Comparison of Arduino Projects and Student Academic Performance

Semester	Number of Circuit Labs	Number of Arduino Projects	The rate of Arduino Projects to Total Experimental	Average Grade in Arduino Projects	Average of Course Final Grade	Average Grade in Labs	Total Number of Students	Students with GPA<2.0	The rate of students with GPA <2.0
Spring 2023	10	7	41%	69%	72.35%	74.63%	22	5	23%
Fall 2023	10	3	23%	69%	75.86%	70%	16	7	44%

The data in Table 1 proposes a potential relationship between the number of Arduino projects and the percentage of students with a GPA below 2.0. Across the two semesters, there was a notable decrease in the percentage of students with a GPA below 2.0 despite an increase in the number of Arduino projects. In Fall 2023, when only 3 Arduino projects were implemented, 44% of students had a GPA below 2.0. Conversely, in Spring 2023, with 7 Arduino projects, only 23% of students fell into this category. Because the other conditions, including the curriculum, have not been changed, and this is the only electrical engineering course offered to freshman students, studying the effect of changes in this course on student outcomes throughout the semester is one of the goals of this research. This correlation suggests that Arduino projects may play a role in improving student academic performance by creating a positive view of the application of Electrical Engineering in their daily life. Moreover, despite variations in average course final grades between the two semesters (72.35% and 75.86%, respectively), the consistent relationship between Arduino projects and reduced low GPAs implies that the impact of these projects transcends fluctuations in overall course performance. This underscores the importance of integrating project-based learning experiences into the curriculum to support student success and mitigate academic challenges. However, this is ongoing research, and it needs to be run and tracked in the following semesters to yield results. There might be other parameters involved in this outcome.

The analysis of Fig. 11 reveals several key findings regarding the correlation between course final grades, Arduino project grades, and students' term GPA for Fall 2023. Firstly, there is a notable association between Arduino project grades and final grades, indicating that students who perform well in Arduino projects tend to achieve higher final grades than those with lower project grades. This correlation is stronger than the relationship between final grades and circuit lab grades.

Moreover, the graph shows that students who excel in Arduino projects are more likely to maintain good standing in the program. Conversely, students with low grades in Arduino projects are at a higher risk of having a GPA below 2.0. Interestingly, this pattern does not hold true for students with high grades in circuit labs but low grades in Arduino projects, indicating that project performance may significantly impact overall academic success. The data also reveals that among students with a GPA below 2.0, their project grade tends to be the lowest average among the three categories (Arduino projects, circuit labs, and final grades). This underscores the importance of Arduino projects in determining student performance and academic outcomes, particularly for those at risk of falling below the GPA threshold.

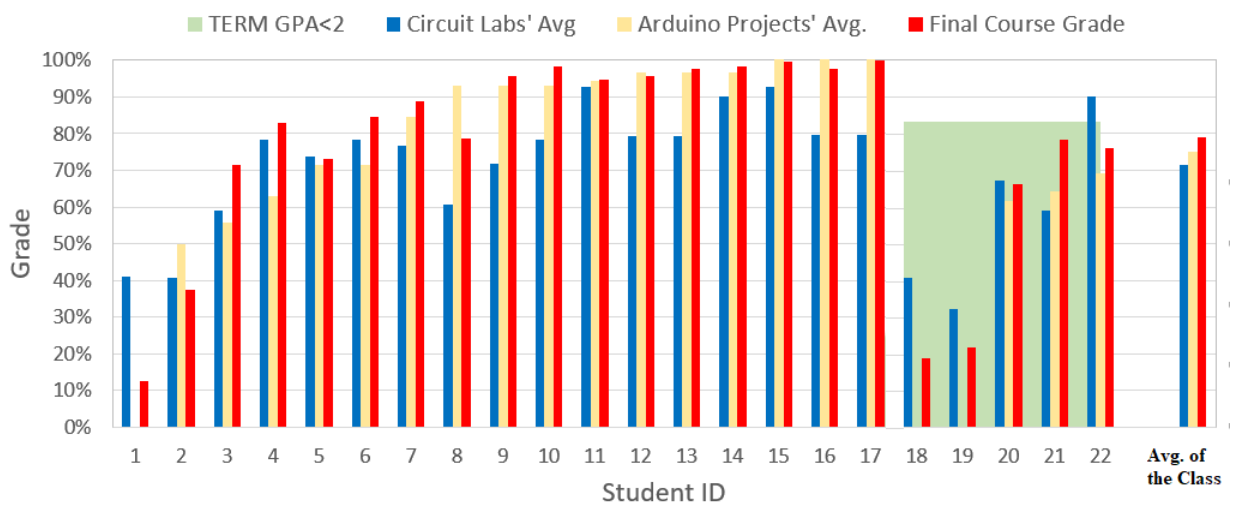


Fig 11. Correlation Between Arduino Project Grades, Final Grades, and Term GPA for Fall 2023

The student evaluations reveal a positive response to the course structure, particularly highlighting the Arduino Projects component. The median score for the criterion regarding using materials and teaching techniques to present content in engaging ways is 5 out of 5. Additionally, student comments consistently express appreciation for the projects, indicating that they feel adequately prepared for further courses in Electrical Engineering.

7. Conclusion

The Freshman Circuits Course serves as a program where EET faculty can interact with freshmen students in their first semester and furnish them with technical experiences geared towards inspiring them to continue with the EET program and begin/continue development of their technical skill base. The Freshman Circuits Course has evolved from being a DC Circuits Introduction to a multifaceted course involving breadboarding, meters, software simulation, Arduino coding, digital circuit projects, and, most recently, some basic digital control of Servo and DC motors. We hope to inspire freshmen with increasingly limited technical backgrounds and math preparations to engage in classroom projects and build their technical knowledge foundation.

The newly added segments involving digital control of Servo Motors and DC Motors have been designed to promote active engagement in the modification and limited design of Arduino code to position Servo Motors and drive DC Motors. During the Servo Motor module, students work on a garage door project, and in the DC Motor module, students move the motor in a rectangle, all through software coding. The motor lab exercises also present linear single-variable algebra problems to the students, for example, students need to perform algebraic calculations to determine PWM timing for DC Motors.

The paper examines student performance across two consecutive semesters, focusing on course activities such as circuit-based lab experiments and Arduino projects. The findings reveal strong correlations between Arduino project grades, final grades, and students' term GPA for Fall 2023, indicating the potential of Arduino projects to influence academic outcomes and student success.

References:

1. Goldberg, S. (2023, March), EET Freshman Circuits Course for the Changing Student Population Paper presented at ASEE Zone 1 Conference - Spring 2023, State College, Pennsylvania. 10.18260/1-2--45063
2. Nasri, Maryam. "Implementing Laboratory and Project-Based Embedded Control Sequence Courses in Electrical and Computer Engineering." 2023 ASEE Annual Conference & Exposition. 2023.
3. Tinker CAD was developed by Autodesk in 2011. Available: https://www.tinkercad.com/
4. Electronics Fundamentals: Circuits, Devices & Applications, 9th edition, Published by Pearson (January 7th, 2021) - Copyright © 2022, Thomas L. Floyd, David M. Buchla, Gary D. Snyder
5. M. Nasri, M. R. Hossain, H. L. Ginn and M. Moallem, "Agent-based real-time coordination of power converters in a DC shipboard power system," 2015 IEEE Electric Ship Technologies Symposium (ESTS), Old Town Alexandria, VA, USA, 2015, pp. 8-13.
6. M. Nasri, M. R. Hossain, H. L. Ginn and M. Moallem, "Distributed control of converters in a DC microgrid using agent technology," 2016 Clemson University Power Systems Conference (PSC), Clemson, SC, USA, 2016, pp. 1-6.