The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #42557

Using Arduino Microprocessors in a Mechanical Engineering Curriculum

Dr. Scott F. Kiefer, York College of Pennsylvania

Scott Kiefer has spent over twenty years teaching mechanical engineering at four different colleges. He started at the University of Puerto Rico at Mayaguez in the traditional role of teaching and administering a modest graduate research program. At Trine University, a small private school in Angola, Indiana, he focused on undergraduate education while teaching ten different courses ranging from introductory freshman courses to senior capstone. Scott also served as an advisor to many different undergraduate research projects. He then moved on to Michigan State University and took a position as a teaching specialist concentrating on undergraduate classroom instruction. Scott finally settled at York College of Pennsylvania. He has been at York College for over ten years and feels as if he has found a place where the focus on teaching and students aligns well with his background and interests.

Dr. Stephen Andrew Wilkerson P.E., York College of Pennsylvania

Stephen Wilkerson (swilkerson@ycp.edu) received his PhD from Johns Hopkins University in 1990 in Mechanical Engineering. His Thesis and initial work was on underwater explosion bubble dynamics and ship and submarine whipping. After graduation he took a

Dr. Ashley J Earle, York College of Pennsylvania

Ashley is an Assistant Professor in the Mechanical and Civil Engineering department at York College of Pennsylvania. She received her B.S in Chemical and Biomolecular Engineering and B.A. in International Studies from Lafayette College. She then pursued h

©American Society for Engineering Education, 2024

Using Arduino Microprocessors in a Mechanical Engineering Curriculum

Abstract

Over the past several years, the Mechanical Engineering Department at York College of Pennsylvania has been using Arduino microprocessors more and more throughout the curriculum. At first, they were only relied upon in one, lab-based course. However, a greater incorporation of the microprocessors into the curriculum has provided a platform for hands-on learning in classes that are more traditionally lecture based. Currently, Arduinos are sometimes introduced in a sophomore level circuits course. All juniors then use the Arduinos exclusively for interfacing with different sensors and actuators in an Instrumentation Lab course. In this course, the microprocessors allow students to design their own experiments to evaluate sensors and to complete a final project of their own design. A senior level Automatic Controls course has also leveraged the Arduinos to learn about different control methods through several different hands-on experiments. The microprocessors allow students to easily change the gains in different types of control algorithms and experience first-hand how the physical response of the system changes. Finally, a senior level elective class in Sustainable Energy has benefited from the use of Arduinos. Students have been able to develop hands-on experiments to explore solar and wind tracking, and measure the power output of alternative energy systems. Repeated exposure to Arduinos through coursework also contributes to student's use of Arduino in classes where not required. The expanded use of the Arduino microprocessors has allowed faculty to enhance learning through hands-on experiences throughout the Mechanical Engineering Curriculum.

1. Introduction

Arduinos were originally conceived of in a classroom in 2005 by a PhD student, Hernando Barragán, under the supervision of Massimo Banzi [1]. The goal of this platform was to provide an affordable electronic platform with a low learning curve that could be used by students in or out of a classroom [2]. In 2010, the Arduino UNO came onto the scene and sparked widespread adoption in many universities across the world. Based on data from Google Scholar, there has been an exponential growth in the number of papers with Arduino, Education, and Engineering as keywords over the last twenty years (**Fig. 1**).

Since modern engineering problems usually include electrical components, developing a working understanding and comfort with microprocessors, sensors, and actuators is necessary for today's mechanical engineers [3]. With this in mind, the Mechanical Engineering Program at York College of Pennsylvania began adopting Arduino into the curriculum in ME351, an Instrumentation lab course. Since that time, a concerted effort has been made to fine-tune and expand the way Arduino is used throughout the curriculum to improve student's comfort with the system.

1.1 Literature Review

Within mechanical engineering, many programs have introduced Arduino in one or more classes, although the timing of Arduino introduction and level of curricular integration varies widely between programs. Most studies have focused on the impact of Arduino usage in a single class with far fewer look at building Arduino throughout a curriculum. Since students use of Arduino is more of a summative measurement, the literature review focused on classes earlier in the curriculum. The field would benefit from a summative literature review of best practices due to the staggering amount of work done in this area. The summary of a brief literature review of classes where Arduino has been implemented at various institutions is shown below in **Table 1**.

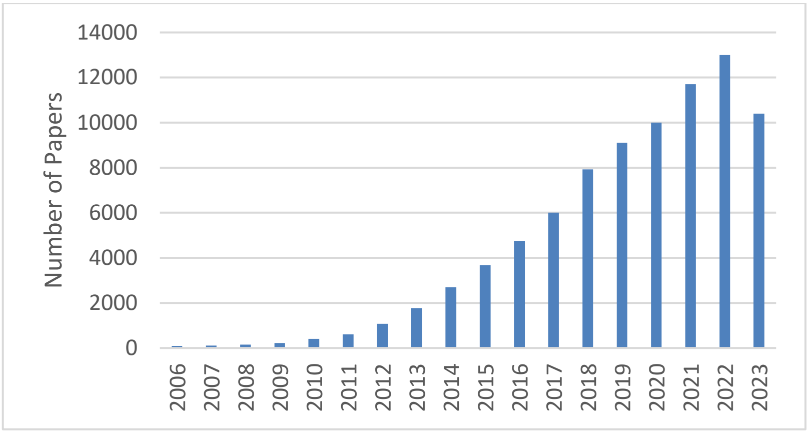


Figure 1. The number of papers published per year in the Google Scholar database that included Arduino, engineering, and education as keywords.

Table 1. Summary of curriculum timing, courses, and Arduino coverage in a subset of literature.

Institution	**Year**	**Courses**	**Req'd**	**Arduino Sensors and Interfaces**	**Ref**
University of Illinois at Chicago	1	Intro to Eng	Y	TinkerCAD IR Sensor, Ultrasonic Sensor Teacher specified project	9
Ohio Northern	1, 2	Intro to Eng Computer Applications Experimental Methods	Y	TinkerCad, MATLAB, LEDs, Motors, etc, Student-led design project	4, 7
South Dakota School of Mines	2	Product Design	Y	LEDs, Motors, Digital/Analog Interface with various shields Student-led design project	3
Tecnológico Nacional de México	2	Mechanics of Materials	Y	MATLAB, Simulink	11
Western New England	3	Mechatronics	Y	LEDs, motors, etc Teacher specified project	5
Santa Clara University	3/4	Mechatronics	Y	LEDs, IR Sensor, Motors, Teacher specified project	10
Prairie View A&M	3	Instrumentation and Measurement	Y	MATLAB, Simulink IR Sensor	8
Science and Technology Fez	3	Dynamics	Y	Motors, various sensors Student-led design project	16
Penn State Berks	4	Vibrations	N	Accelerometer Comparison to theory + FEA	6

One of the key themes in the literature is using the Arduino platform to replace Labview, especially for instrumentation/experimental design classes [4, 8]. A major benefit to the Arduino platform is low cost, which allows students to choose from a wide range sensors as opposed to having very limited options. This low-cost element of Arduino typically does reduce the accuracy and lifespan of the sensors, however, this opens an additional area for students to discuss accuracy and precision [4]. Another overarching theme is that the students have better feedback for the Arduino based classes compared to standard lecture, although appropriate scaffolding needs to be in place to prevent students' frustration and excessive workload [3, 4, 6, 10, 16].

This paper describes curriculum modifications and student responses to its inclusion in various classes, summarized in **Table 2**. Similar to the work done at Prairie View A&M, the Instrumentation course that had traditionally used Labview as the platform for exploring instrumentation and microprocessors was modified to instead use the Arduino microprocessor several years ago [8]. This platform provided a good place to introduce many different sensors, and provided data that could be used to teach statistical analysis while reducing student wait time to use the limited Labview equipment. Having students that had already used the Arduinos opened the door for an Automatic Controls course to use them to provide some hands-on PID controls activities. It also helped to provide a way for students to control the position of solar panels in a Sustainability course.

Table 2. Summary of implementation of Arduino through the curriculum at XXX

Year	**Courses**	**Req'd**	**Arduino Sensors and Interfaces**
3	Instrumentation and Microprocessor Lab	Y	IR Sensor, Ultrasonic Sensor, Motors, Digital/Analog Student-led design project
4	Controls	Y	MATLAB PID controls, various sensors
4	Sustainability	N	MATLAB Motors, voltage sensors

2. Arduinos in an Instrumentation Laboratory Course

The first course to consistently rely heavily on the Arduino microprocessors was an instrumentation laboratory course. Prior to 2020, the instrumentation class had used Labview to complete activities using different sensors provided by National Instruments. The course consisted of students completing lab activities each week evaluating the performance of one sensor, and then completing a lab write up forming conclusions about the sensor using statistical analysis of the data collected.

The Arduinos provided a platform that allowed students to interact with a wider variety of sensors and actuators in the physical world. They also allowed the course instructor to provide a laboratory preparation guide, including support material, to prepare students for the course activities before coming to the classroom. The students would then come to class ready to complete a project using a variety of sensors and actuators working together. In addition, some weeks were used to have students design their own experiments to evaluate the use of sensors in different applications.

2.1 Sensors and Actuators in Instrumentation Lab

The ABET course outcomes for the Instrumentation lab course include exposure to electrical and electro-mechanical devices including various sensors, actuators, and instrumentation used in electrical and mechanical applications. The Arduino platform is a great base to use to provide activities where students can interface sensors and actuators with the physical world. Lists of the components used and the learning outcomes covered are below in **Table 3** and **4**.

Table 3: Components Used in Instrumentation Lab

Components Used in Weekly Activities	
Visible LEDs	Potentiometers
Switches (push buttons)	Transistors
Infrared LEDs	DC Motors
Infrared Transistors	Stepper Motors
Seven-segment Displays	Servo Motors
Ultrasonic Distance Sensors	H-bridges (motor drivers)
Operational Amplifiers	Temperature and Humidity Sensors
Diodes	Current and Voltage Sensors

Table 4: Learning Outcomes in Instrumentation Lab

Learning Outcomes Covered in Weekly Activities
Measuring Voltage and Current with a Multimeter
Limiting Current and Voltage in a Circuit
Reading a Switch Position
Debouncing a Switch
Using Functions in Programming
Keeping Time with a Microprocessor
Calibration of Ultrasonic Sensors
Digital to Analog Conversion
Kirkoff's Voltage and Current Laws
Voltage Dividers
Amplification of Current Required to Drive Motors
Controlling Rotation of a Stepper Motor
Controlling Position of a Servo Motor
Using Arduino Libraries

2.2 Teaching Method

Using the Arduinos, along with course management software, allowed the course instructor to give students fairly extensive preparation before coming to the lab each week. Because there is a plethora of information available for Arduinos online [13, 14], a handout and video links were provided for each week's lab. Students were also given an online quiz to help ensure that they had completed the preparation material before coming to class. This allowed the students to begin building the circuits and writing the code for each assignment without a lot of introductory lectures. The instructor could then circulate among the students and help trouble shoot any problems as the students worked through the activities.

The instructional handouts were scaffolded so that they started with a lot of information including things like example code, and pictures of constructed circuits. As the instructions continued, and the students gained confidence, the code examples disappeared. Circuit diagrams were now given without pictures of the actual construction. Each week concluded with example project assignments for the students to complete on their own where they needed to apply the concepts that were covered in the handout without further instruction. An example of one of the weekly assignments is given in **Appendix A**.

After 11 weeks of structured activities with different sensors and actuators, the students were assigned a final project using the Arduinos. They created their own projects which included a project proposal to be approved by the instructor. Students were allowed to use some of the sensors and actuators that had been used during the semester, but they also needed to do their own background research to determine how to use sensors that they were not already familiar with.

2.3 Teaching Statistical Methods with the Arduino

An additional ABET required outcome of the instrumentation lab course was that students be able to perform statistical analysis of laboratory data to form conclusions. Having the Arduinos in the course provided a great tool for collecting the data to be analyzed. Students were given three assignments, based on real life scenarios, where they would use the Arduinos to collect data and perform statistical analysis to form conclusions.

The first scenario involved determining if a specific infrared sensor could be used to determine the difference in plastics of different thickness and opacity. Students were given several different groups of plastics pieces each with slightly different properties. They then needed to design and conduct experiments to determine if their sensor could detect a statistical difference in each group. To help prepare them for the task, an example was first done in the lab measuring the length of two sets of Legos bricks and demonstrating how to look for a statistically significant difference in the two groups [15].

The second statistical exercise involved using an ultrasonic sensor to measure distance. Students were first shown how to construct a calibration curve, a linear regression, and a residual plot using some data that was provided to them in a class exercise. Then, they were tasked to create their own calibration curve for an ultrasonic sensor measuring the distance to an object of their choosing. They were required to determine if their data was linear and what the best fit for their calibration curve would be. Then, they needed to conclude the maximum distance the sensor could be used, and what was the predicted possible error.

The final statistical exercise involved measuring the power output of a wind turbine. The statistical analysis was very similar to that of the distance sensor, so there was no new class activity involved. Students were simply asked to create and analyze a calibration curve for a sensor that measured current and voltage to determine power output of a motor that was being driven by a "turbine". The wind "turbine" was simulated using a simple DC motor connected to a variable power supply.

2.3 Student Response to Arduino in Instrumentation and Microprocessor Laboratory

The use of Arduinos in the Instrumentation Lab course has received positive reviews in the course evaluations from students over the three years they have been used as a foundation for the course. The number of respondents decreased each year from n_{21}= 27 (60% of students), n_{22} = 17 (37% of students), to n_{23} = 11 (26% of students). While there is not a lot of power to the data, some trends can still be determined. The first year the course was taught with Arduino, there was a heavier emphasis on writing and statistics which was adjusted in future years to reduce workload to compensate for pre-lab work and the extended time needed for coding during lab time (**Fig. 2**). The first year had fewer students saying that the Arduino, sensors/circuits, and coding were most valuable and more students saying that writing/statistics were most valuable. However, in the second iteration, no students indicated a high value on writing or statistics whereas the value for the technical elements of the class went up two to three times from 2021. In 2023, a better balance was reached so that both the technical and professional elements of the course seemed valuable to the students.

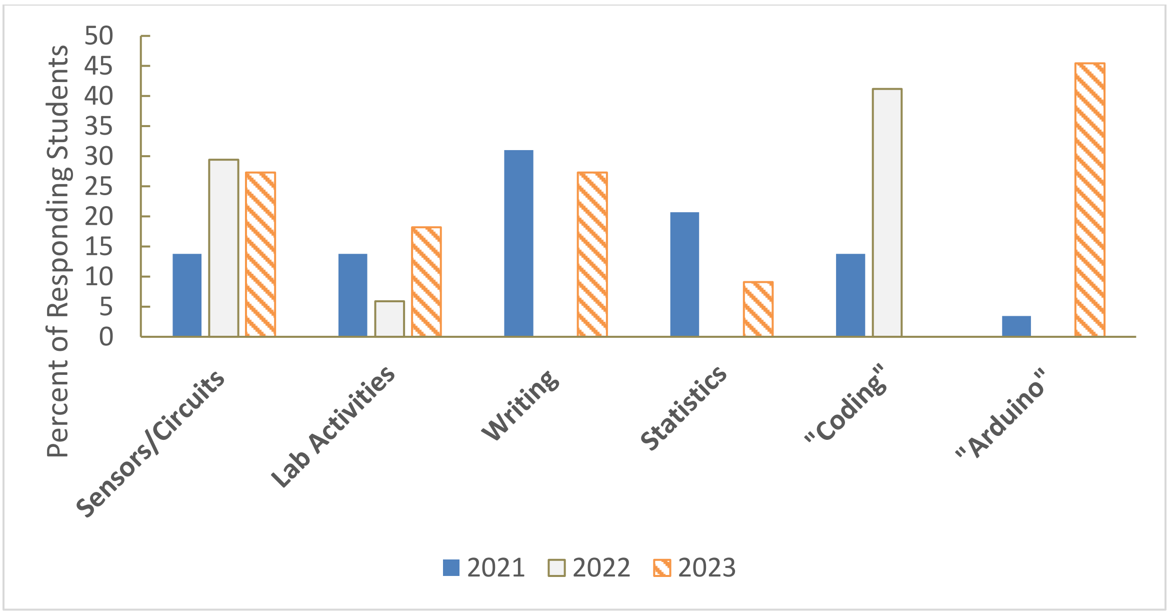


Figure 2. Percent of students describing a particular course element as "most valuable". The number of students responding each year to the course evaluation were n_{21}= 27, n_{22} = 17, and n_{23} = 11.

As the class evolved, another interesting take away was how the students described the technical aspect of the course trending from the general to the specific. For the first two iterations, students used words such as “sensors”, “circuits” and “coding”, but in the most recent iteration students focused much more on the Arduino platform compared itself, using words and phrases such as “microprocessor”, “Arduino”, and “Arduino functions”. Only one student from 2021-2022 mentioned Arduino specifically in their comments compared to nearly half of the students naming Arduino in 2023.

In terms of implementation, there were several similar responses to learning coding as found in other institutions. A few students complained about coding being frustrating. One student said, “There's sometimes just a difficulty in getting code to work despite following the proper procedures and it can really be annoying.” This is a common frustration mentioned throughout the literature regardless of the class year where Arduino’s were introduced [9, 16]. However, as with other institutions, it was determined that having willing and available faculty/TA help brings the frustration to a reasonable level. In the first implementation, 10% of students complained that the instructor spent too long with one group, leaving other groups to wait too long to receive help. Updating the structure and the way in which help was given prevented these comments in subsequent years.

The other comment that came up many times in the literature review was having the appropriate scaffolding to help students achieve success with coding (without giving them the code) helps students derive value from the experience [6]. In 2021, there were some later classes where the scaffolding was too much. As indicated by a student, “In some of the later classes, we were just given code to more or less copy and paste and be done with,” which led them to say the coding was the least valuable aspect of the course. Another student added, “I think there were times when a nudge in the right direction would have been better than telling us exactly what to fix right away. Could be done by re-explaining what a command is used for or something like that.” As the instructors had more practice with the implementation, these comments were fewer as the correct level of scaffolding was reached. In 2022, in response to being asked what the most valuable aspects of the course were, a student noted, “The assignments where we were given the basics and then had to work to figure out more complex ways of applying that with previous knowledge was great.”

Interestingly, there were also two students in 2021, the first implementation, who specifically mentioned that they did not understand how coding was relevant to mechanical engineers. Anecdotally, this is something that students have said before in the program. Interestingly, those comments did not appear in 2022 or 2023. It is hard to tell whether the students are experiencing more instances where the electronic integration is made to feel relevant through Co-op or other experiences, or whether the course instruction has changed.

One important note is that some students found the class less valuable due to similarities to a Mechatronics course during the second year. Currently there is a course taught by electrical engineering faculty to mechanical engineering students in the spring of their sophomore year. This course has sometimes incorporated the use of Arduinos, depending on the course instructor. Some instructors use the Arduinos throughout the semester, and some teach the course without microprocessors, more like a traditional circuits class. The class does have a lab component, so students in the more traditional circuits class only become proficient at constructing circuits and analyzing them with multimeters and oscilloscopes. Students in other sections used the Arduinos throughout the course. Students who had taken the Mechatronics course with the Arduinos said that Instrumentation Lab was, “essentially Mechatronics 2.0. [We] did almost the same thing that we did in mechatronics.” Another student said, “Nothing [was valuable] until the final project as most [of the material] was covered in mechatronics.” In total over all three years of the new class, 5/57 of the responding students indicated this as an issue.

3. Automatic Controls use of the Arduinos

Knowing that students who were in the Automatic Control course were familiar with the Arduinos and programming, the instructor built several PID instructional devices (one is shown below in **Figure 3**). This device has a track on top for a steel ball to roll along. The bottom of the track is lined with a linear potentiometer that changes resistance as the ball rolls. The resistance is read by the Arduino, which in turn rotates the servo motor to change the angle of the track. The task is to program the Arduino using a PID controller so that the ball stays in the center of the track. Other devices such as a levitating ping-pong ball and rotating pendulum were also used during the class.

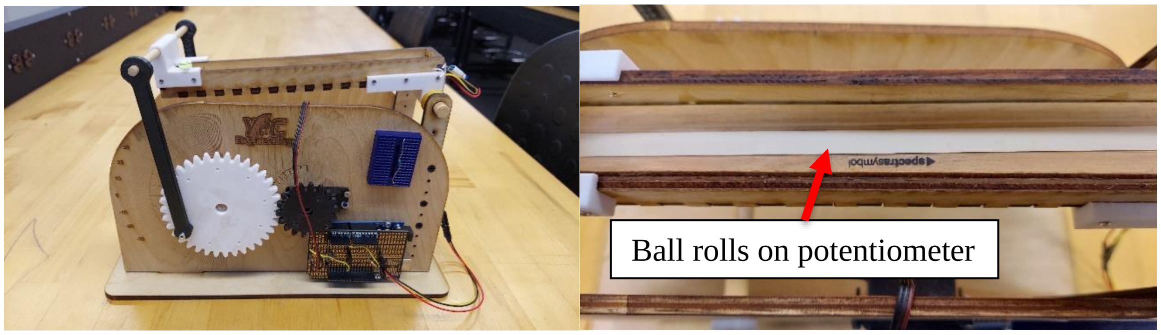


Figure 3: PID Demonstration Device Used in Automatic Controls

Because the students were already familiar with the Arduinos, the instructor was simply able to demonstrate a working example for the class and give the students a few hints about how to write the code for a PID controller. The students were then able to get the devices working and alter the gains for the proportional, integral, and differential to demonstrate the effects of each control effort.

4. Arduinos in a Sustainable Energy Course

Similar to the use of Arduinos in Automatic Controls, they were also used an elective Sustainability course. The class was studying the effects of the angle of the sun on the efficiency of a solar panel. A small solar panel was set up an Arduino was used to measure the power output of the solar panel. It was then used to rotate the panel to follow the sun as it moved through the sky so that the power generate could be compared to the stationary panel.

5. Arduinos in Non-required Projects

Since the Arduinos have become readily available throughout the engineering facilities, and because the students have become comfortable using them, they have appeared in many different student projects throughout the curriculum. Students have used them in Machine Design projects to do things like measure the effectiveness of a tourniquet and to measure pressure and count the output of a pellet press. They have also appeared in Capstone projects such as a prosthetic hand project, electrospinning project, and the SAE formula car. While no historic data was collected prior to the first implementation, 11/41 students (26.8%) from the second implementation of the Arduinos in Instrumentation Laboratory indicated that they have used Arduino since the course in a context in which it was not required. In comparison, only 7/53 (13.2%) students in their freshmen year have ever used an Arduino at all, and none of these students would have been comfortable using them again without help.

It should be noted that Labview was used in the instrumentation class for at least eight years prior to the introduction of the Arduinos. While there was not any official assessment done during this time, none of the students who had taken the class during this period had ever used Labview in any projects outside of the instrumentation class. In fact, many of the students who had taken the class did not remember that they had used Labview at all when it was mentioned in casual conversation.

6. Recommendations for Future Applications

After seeing the success and proliferation of the Arduinos in the Mechanical Engineering program, the next logical step would be to include them a little earlier in the curriculum. Based on other literature, Mechatronics is a popular class to include Arduino based education [5,10]. Since there is a version of a Mechatronics course that includes the Arduino, bringing this curricular element back into the sophomore year would allow for more advanced work to be done in Instrumentation Lab. The Instrumentation Lab course could then be modified to include more advanced projects with multiple sensors and actuators working together. In addition, manufacturing environments could be simulated with appropriate industrial components and PLCs could be introduced.

Another possible place to introduce mechanical engineering students to the Arduinos is a first semester freshman introduction to engineering course. This course is currently taken by electrical, mechanical, and civil engineering students, and has time set aside to explore all three of the disciplines. The Arduinos have made an appearance in this course in the past, but have not been used in the past several years. Exposing first year students to Arduino requires more scaffolding, as indicated by groups at Ohio Northern and University of Illinois at Chicago [4,7,9]. While York College of Pennsylvania has small class sizes, which improves students access to faculty help, a full-scale implementation may require additional support from older students in a TA role to be completely successful. Another barrier to implementation is that the faculty instructing this course rotate on a frequent basis. As not all faculty are comfortable with Arduino, it would require whole department agreement that these skills are valuable enough to be placed in the class long term. In addition, if Arduino was introduced in the first year, it would likely be beneficial to find ways to incorporate it into each of the following semesters to provide the deepest benefit to the students.

Right now, there is relatively limited data on the impact of the increased presence of Arduinos in the curriculum at York College. Introducing a survey based on skill and interest with mechatronics concepts, or even Arduino in particular, at the end of every academic year would improve the robustness of the data. Since most of the literature focuses on the impact of Arduino in a single course, or even a single activity, tracking the impact of increased presence across the curriculum would add a lot of value to the community.

8. Summary and Conclusions

The application of Arduinos in a mechanical engineering curriculum has been very successful. It has provided a platform that students use to explore different sensors and actuators in an instrumentation course. It has also allowed students to be able to design their own experiments and use the Arduinos in a design project that they develop. In addition, the Arduinos have given faculty an opportunity to provide more hands-on activities for students in controls and sustainability courses. Finally, because students have become comfortable with the Arduinos, they have begun to choose to use them in design projects in many additional courses. As Arduino use continues to expand across the mechanical engineering curriculum, the effects on student skills and tendencies will be systematically tracked.

The freshman course sequence for all disciplines of engineering students at York College is currently being revised. One of the considerations of the development team is whether Arduinos should have a permanent place in the first semester design course. This course will include a design project, and it would be a great place to introduce students to some of the functionality of the Arduinos.

In addition, the authors of this paper have requested that Arduinos be included in the Mechatronics class that students take in their second year, regardless of the course instructor. If all sections of this course were expanded to include using the Arduinos to control some basic sensors and actuators, then the instrumentation course could expand its statistical analysis activities, include more advanced sensors projects, and expand the scope of its design project.

Since students are already naturally choosing to use the Arduinos for projects in their higher-level courses, there is not a need to specifically expand Arduino activities in upper-level classes. However, having all the students familiar with the Arduino programming does make it easy for instructors to continue to add educational activities that use the Arduinos to their courses.

9. Bibliography

[1] Sam, "History of Arduino - Tutorial Australia," *Core Electronics*, Apr. 12, 2022. Available: https://core-electronics.com.au/guides/history-of-arduino/#:~:text=The%20very%20first%20Arduino%20board. [Accessed: Jan. 30, 2024]

[2] ArduinoTeam, "One board to rule them all: History of the Arduino UNO," *Arduino Blog*, Dec. 09, 2021. Available: https://blog.arduino.cc/2021/12/09/one-board-to-rule-them-all-history-of-the-arduino-uno/

[3] Mark David Bedillion, Karim Heinz Muci-Kuchler, and Walelign Messele Nikshi, "An Arduino-Based Hardware Platform for a Mechanical Engineering Sophomore Design Course," *Papers on Engineering Education Repository (American Society for Engineering Education)*, Sep. 2020, doi: https://doi.org/10.18260/1-2--29774

[4] L. W. Funke, J. Blake Hylton, and D. Sawyers, “Work in Progress: Incorporating Microprocessors across the Mechanical Engineering Curriculum,” *ASEE Conference Proceedings*, Sep. 2020, doi: https://doi.org/10.18260/1-2--33630

[5] Jose Antonio Riofrio and S. G. Northrup, “Teaching Undergraduate Introductory Course to Mechatronics in the Mechanical Engineering Curriculum Using Arduino,” *ASEE Conference Proceedings*, Sep. 2020, doi: https://doi.org/10.18260/1-2--22539

[6] Joseph Michael Mahoney and R. Nathan, “Mechanical Vibrations Modal Analysis Project with Arduinos,” *ASEE Conference Proceedings*, May 2018, doi: https://doi.org/10.18260/1-2--28660

[7] D. R. Mikesell and J.-D. S. Yoder, “Introducing Mechanical Engineers to Microprocessors with Arduino Tank Robots,” *Papers on Engineering Education Repository (American Society for Engineering Education)*, Jul. 2015, doi: https://doi.org/10.18260/p.24362

[8] M. Han and C. Duan, “Different Methods of Programming for Mechanical Engineering Students: A Case Study,” *Proceedings of ASME*, Nov. 2019, doi: https://doi.org/10.1115/imece2019-11424

[9] J. Szwalek, Y. Siow, and Jaqueline Rojas Robles, “Design Course in a Mechanical Engineering Curriculum,” *2020 ASEE Virtual Annual Conference Content Access Proceedings*, Sep. 2020, doi: https://doi.org/10.18260/1-2--34394

[10] R. S. Grover, S. Krishnan, T. E. Shoup, and M. Khanbaghi, “A competition-based approach for undergraduate mechatronics education using the arduino platform,” *Fourth Interdisciplinary Engineering Design Education Conference*, Mar. 2014, doi: https://doi.org/10.1109/iedec.2014.6784685

[11] Manuel Alejandro Ojeda-Misses and Carlos Dávila Chavero, “Educational Platform Based on a Mechanical Beam With Performance in Real Time,” *Revista Iberoamericana De Tecnologías Del Aprendizaje*, vol. 18, no. 3, pp. 258–267, Aug. 2023, doi: https://doi.org/10.1109/rita.2023.3301413

[12] Mechanical Engineering Curriculum, York College of Pennsylvania, http://catalog.ycp.edu/preview_program.php?catoid=40&poid=4203

[13] *Language Reference* (2024). Available at: https://www.arduino.cc/reference/en/

[14] *Uno R3 – Getting Started* (2024). Available at: https://docs.arduino.cc/hardware/uno-rev3/

[15] Bodnar, Cheryl, and Kaitlin Mallouk. 2018. "GRASPS: Goal, Role, Audience, Situation, Product, Standards - Framing Any Assignment for Curiosity and Connections". Engineering Unleashed. Tuesday, October 30, 2018. https://engineeringunleashed.com/card/1444

[16] Jihane Kojmane and A. Aboutajeddine, “Enjoyeering Junior: A hands-on activity to enhance technological learning in an engineering dynamics course,” *IEEE Explore*, Mar. 2016, doi: https://doi.org/10.1109/it4od.2016.7479267

Appendix A – Example Weekly Assignment from Instrumentation Lab Course

Buttons, Timing, and Numeric Displays

Assignment Goals

At the end of this assignment you should be able to:

1. Connect the terminals of a seven-segment display to the output pins of an Arduino and program the Arduino to display the numbers from zero to nine.
2. Demonstrate the use of functions in programming the Arduino.
3. Apply the timing functions that are included in the Arduino software.
4. Demonstrate the debouncing of a switch.

Problem Assignments

1. Design and build a circuit using the Arduino that will count from zero to nine seconds and display the result on a seven-segment display.
2. Design and build a circuit that will count the number of times a button is pressed in 10 seconds and display the result to a seven-segment display.
3. Use the circuit from problem 2, and display the time between two button presses.

The Seven-segment Display

A seven-segment display is really just seven diodes arranged in a pattern so that it can be used to display the numbers from zero to nine (see figure 1). The seven-segment display in your kit is known as a “common cathode” display. This means that all the segments on the negative side of the diode are connected to one common pin which we will connect to ground through a resistor. Then, each of the LED segments can be illuminated by supplying 5V to the positive side pin that corresponds to the segment we wish to light (see the circuit diagram given in figure 1). Because each of the segments is a diode, you must limit the current passing through it to around 10 mA to avoid damaging it. Therefore, we must have a 220 Ω resistor somewhere in our circuit to limit the current passing through each of the diodes (just like we did in Module #2). The easiest way to get 220 Ω of resistance into all the circuits is by putting it between the common cathode pin and ground as shown in figure 1. You **must include the 220 Ω resistor** in the circuit or you will destroy the seven-segment display.

As a part of project #1, we will be writing separate functions for each number from zero to nine. This way when we want to display a number all we have to do in call the function for that number. You will find good documentation on functions at:
https://www.arduino.cc/en/Reference/FunctionDeclaration.

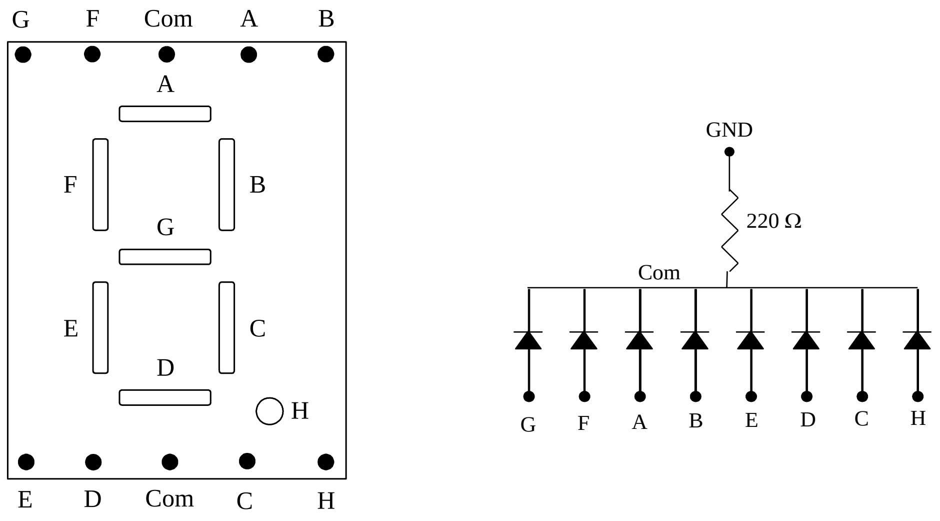


Figure 1: The Seven-Segment Display Pin Connections

Procedure for Connecting the Seven-segment Display

1. Insert the seven-segment display into your breadboard across one of the ridges so that each side of the display is in a separate row of connected holes (see figure 2).
2. Connect the center pin of either the top or bottom (the center pins on the top and bottom are connected to each other internally) of the seven-segment display through a 220 Ω resistor and connect it to GND. (This will serve as the "common cathode" and provide a resistance and ground connection for all of the LEDs.) (also shown in figure 2)
3. Temporarily connect the upper left pin (labeled G in figure 1) of the seven-segment display to 5V. This should light the center LED of the display (labeled G in figure 1).
4. Remove the side of the wire you just connected to 5V and reconnect it to one of the 14 available digital input/output (I/O) pins of the Arduino. If you have forgotten where the I/O pins are located, look back at Module #2. You can use any of the 14 pins (labeled 0-13), but make a note of which pin you are now connected to so you will know which Arduino pin controls this segment (example G – pin 8).
5. Now write a simple program for the Arduino to turn the center segment of the seven-segment display on. You will need to use pinMode() to set your pin as an OUTPUT pin in the setup() section. Then you can set the pin to HIGH or LOW in the loop() section.
6. Repeat steps 2-4 for each of the seven connections of the display. Remember to record which Arduino pin is controlling which segment of the display. You could also set some integer variables at the top of your program mapping each pin number to the letter of the display segment.
7. You should now be able to display any number from zero to nine by setting the proper pins of the Arduino to LOW (0V) or HIGH (+5V).

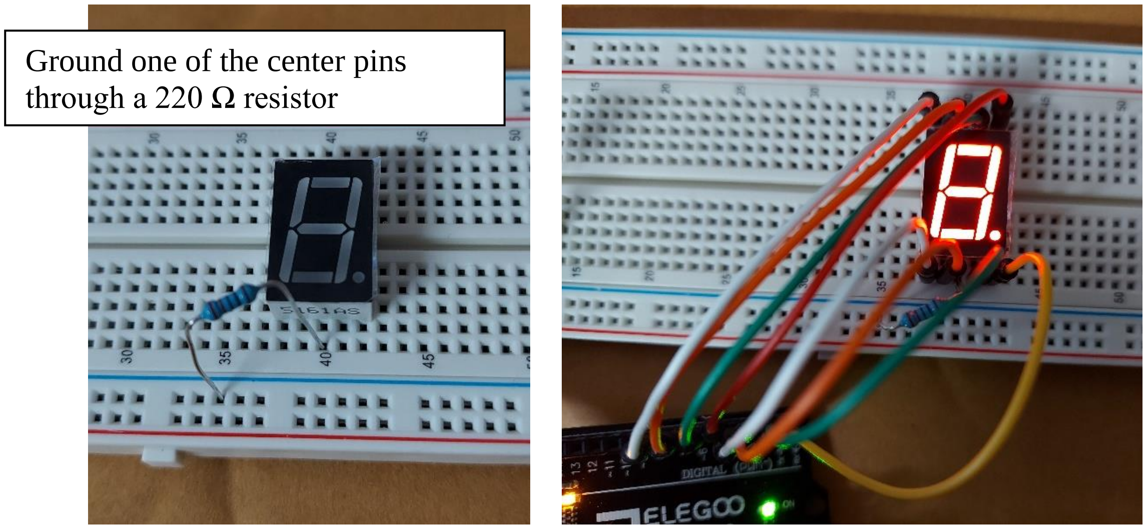
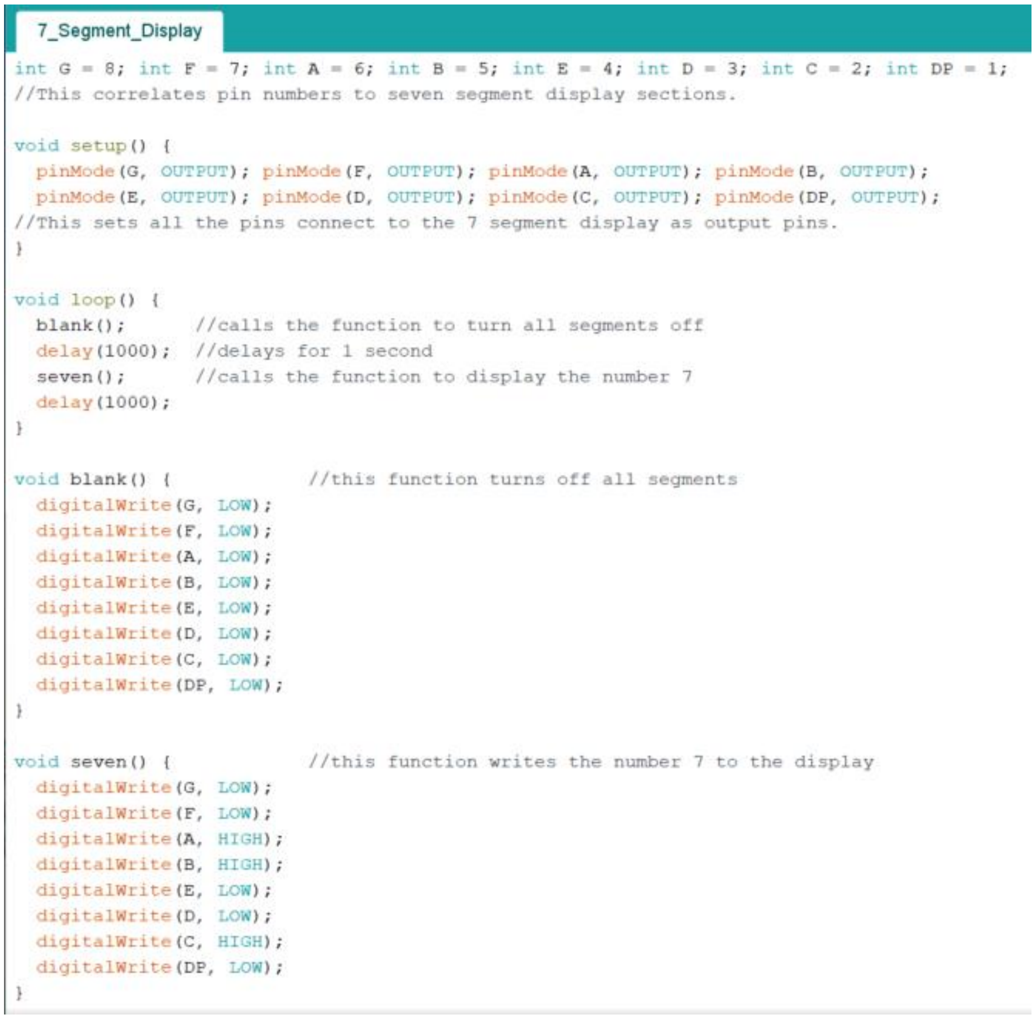


Figure 2: Connecting the Seven-segment Display to the Arduino

Writing Functions to Count from Zero to Nine (Project 1)

Now that we have each of the LEDs of the seven-segment display connected to the output pins of the Ardiono, we will write separate functions to display each of the numbers. There is good documentation on how the Arduino uses functions at: https://www.arduino.cc/en/Reference/FunctionDeclaration. Functions will be written in your code after the “void loop()” section. They will also start with “void” (if you are not passing any variables back from the function), then the name of the function and parentheses. All the commands that are to be completed inside the function will then be included inside brackets, {}. You will call the function from within the “void loop()” section. The function will run through all the lines of code one time and then return to the place where it left the “void loop()” section. An example of using a function to output the number 7 is given in figure 3.



```
int G = 8; int F = 7; int A = 6; int B = 5; int E = 4; int D = 3; int C = 2; int DP = 1;
//This correlates pin numbers to seven segment display sections.

void setup() {
  pinMode(G, OUTPUT); pinMode(F, OUTPUT); pinMode(A, OUTPUT); pinMode(B, OUTPUT);
  pinMode(E, OUTPUT); pinMode(D, OUTPUT); pinMode(C, OUTPUT); pinMode(DP, OUTPUT);
//This sets all the pins connect to the 7 segment display as output pins.
}

void loop() {
  blank();        //calls the function to turn all segments off
  delay(1000);    //delays for 1 second
  seven();        //calls the function to display the number 7
  delay(1000);
}

void blank() {              //this function turns off all segments
  digitalWrite(G, LOW);
  digitalWrite(F, LOW);
  digitalWrite(A, LOW);
  digitalWrite(B, LOW);
  digitalWrite(E, LOW);
  digitalWrite(D, LOW);
  digitalWrite(C, LOW);
  digitalWrite(DP, LOW);
}

void seven() {              //this function writes the number 7 to the display
  digitalWrite(G, LOW);
  digitalWrite(F, LOW);
  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(E, LOW);
  digitalWrite(D, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(DP, LOW);
}
```

Figure 3: Code to Display the Number 7

To save time and frustration, it will work best if you write one function and download it to make sure the code is working correctly. Once you have one working correctly, then move on to another function. After you have all of the functions completed (for numbers 0 to 9), put all the calls into the “void loop()” section separated by one second delay commands. Your Arduino should now be counting from zero to nine with a one second delay between each number.

It should be noted that 7-segment displays are more often controlled by setting-up the data to send to the display as an array rather than turning on each segment individually. Because 7-segment displays are commonly used, there are libraries that you can add to the Arduino programming language to use a series of commands to set up the display and automatically set each of the segments to the correct value. You will notice that your kit has a 4-digit, 7-segment display that only has 12 pins. If you are interested, you will find a good example of how to use this 4-digit display at the link below:

https://www.instructables.com/Using-a-4-digit-7-segment-display-with-arduino/

3.1 Debouncing a Switch

Project 2 will ask you to count the number of times a button is pressed. When you press a momentary contact switch (such as the buttons we are using), the opening and closing of the circuit is not as clean a process as you might think. As you are pushing the button and the electrical contacts are getting closer and closer together they may actually make contact and open again several times before making a final connection. Think of what happens when you drop a ping pong ball a short distance to a very hard surface. It will bounce very rapidly several times before the motion stops. Unlike the ping pong ball, the closing of a switch will rebound several times in a very short time (a few microseconds). Because the Arduino can read changes that fast, it may read three or four changes when you push a button only once.

In order to compensate for the rebounding of the button as it is pressed, we must put in a short delay (0.01 seconds) when we are reading the position of the button. This is commonly referred to as debouncing.

3.2 Keeping Time with the Arduino

In order to complete project 2, we also need to learn about how the Arduino keeps track of time. The small metal elliptical shaped component that is mounted on your Arduino board (show in figure 4) is a crystal. A crystal is a device that oscillates at a certain frequency when a voltage is applied to it (think of it as a controlled vibration). The Arduino can keep track of time using the oscillations of the crystal.

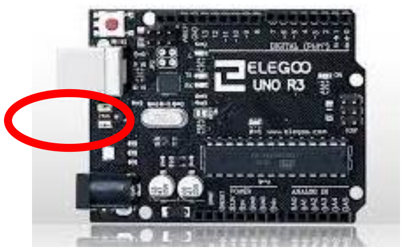
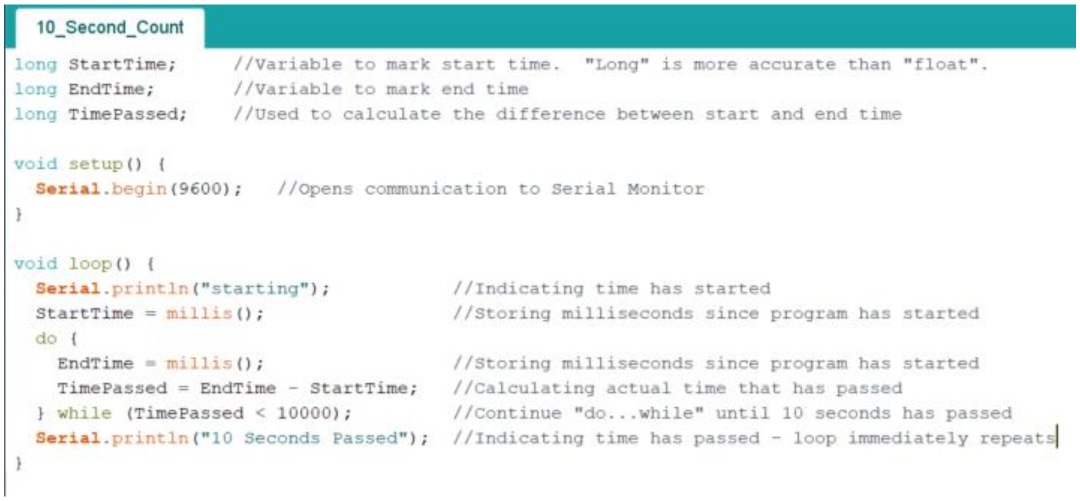


Figure 4: Crystal Mounted on Arduino Board

Using the crystal, the Arduino automatically keeps track of the time that has passed since the current program started to run. You can recall this time (in milliseconds) at any point using the "millis()" command. If you want to measure the time that it takes for a certain event to happen, you will set a variable for the start time using the 'millis()" command. Then, set another variable for the end time and subtract the two values to determine how long an event takes to complete. The is a good description of the "Millis()" command in https://www.arduino.cc/reference/en/.

For Project 2, we will use a "do…while" loop to determine when 10 seconds has passed. The sample code given below in figure 5 is an example of how it can be done. The program will use the Serial Monitor to display the word "starting" when the loop begins and "done" when 10 seconds has passed.



```
long StartTime;      //Variable to mark start time.  "Long" is more accurate than "float".
long EndTime;        //Variable to mark end time
long TimePassed;     //Used to calculate the difference between start and end time

void setup() {
  Serial.begin(9600);   //Opens communication to Serial Monitor
}

void loop() {
  Serial.println("starting");          //Indicating time has started
  StartTime = millis();                //Storing milliseconds since program has started
  do {
    EndTime = millis();                //Storing milliseconds since program has started
    TimePassed = EndTime - StartTime;  //Calculating actual time that has passed
  } while (TimePassed < 10000);        //Continue "do...while" until 10 seconds has passed
  Serial.println("10 Seconds Passed"); //Indicating time has passed - loop immediately repeats
}
```

Figure 5: Code for Measureing 10 Seconds

3.3 *Counting the Number of Times a Button is Pressed (Project 2)*

You should now have all the necessary tools to count the number of times a button is pressed in 10 seconds.

Use the same switch circuit that we used in Module #2 for the button input to the Arduino. You will again need to use the “digitalRead()” command to determine if the switch is being pressed or not. This program is more difficult than Module #2 because you need start a timer and remember the number of times the button is pressed before the time expires.

Because you are already using a “do…while” loop to keep track of the time, you can check the state of the button (pressed or not) each time through the loop. I suggest using three variables: 1) an integer to remember the number of presses (let’s call it “Count”), 2) a boolean (bool) to remember the state of the button the last time through the loop (let’s call it “LastState”), and 3) a boolean to read the state of the button the current time through the loop (let’s call it “PinState”). A boolean is simply a variable that can be either true (HIGH) or false (LOW). You can find more information in: https://www.arduino.cc/reference/en/.

The procedure is then as follows (this is also shown in the flow chart in figure 6):
1) Initialize Count to zero and LastState to false before the Do Loop
2) Inside the Do Loop, set PinState to the value of the button (true if pressed)
3) If PinState is different than LastSate then add one to the count
4) Set LastState equal to PinState
5) Put in a delay of 0.01 seconds (delay (10)) to debounce the button
6) Repeat the loop

After 10 seconds have passed, your program should exit the “do…while”. The only problem is that Count will actually be twice the number of times the button has been pressed. (It indexes by one when the button is pressed and by one more when the button is released). Use integer division to divide Count by 2 and then write the result to the seven-segment display.

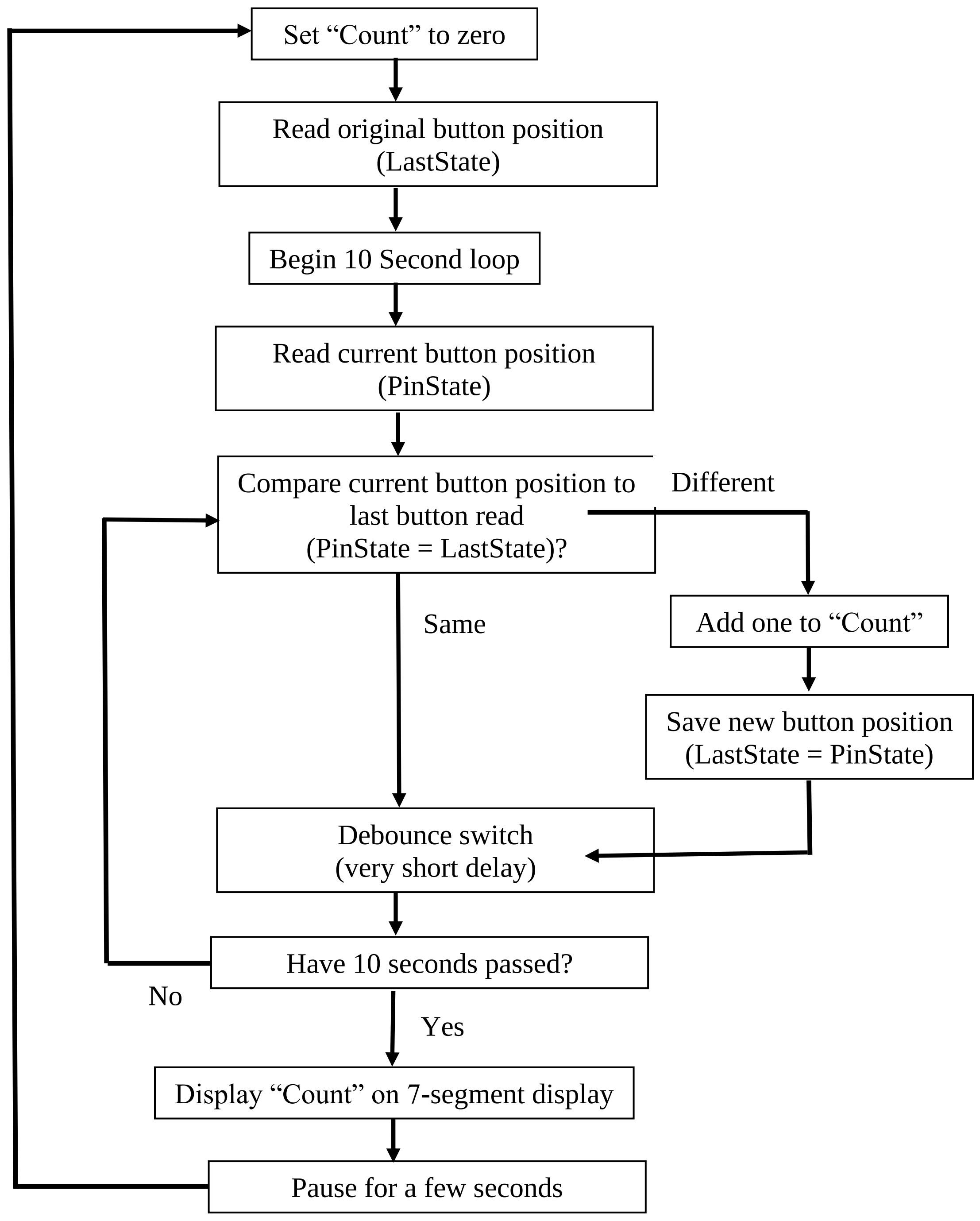

Set “Count” to zero
Read original button position (LastState)
Begin 10 Second loop
Read current button position (PinState)
Compare current button position to last button read (PinState = LastState)?
Different
Add one to “Count”
Save new button position (LastState = PinState)
Same
Debounce switch (very short delay)
Have 10 seconds passed?
No
Yes
Display “Count” on 7-segment display
Pause for a few seconds

Figure 6: Flow Chart for Button Counting Program

3.4 Measuring the Time Between Button Presses (Project 3)

Now that you have an understanding of how the Arduino keeps time, use the same button circuit and 7-segment display circuit you have been using to display the number of seconds between two button presses. To display the number of seconds using your 7-segment display, you will have to find a way to round the actual value of time to the nearest integer between zero and nine. A couple of ways to do this are by using the modulus operator (%), or by converting the variable type from a real number to an integer. Use the Arduino reference documentation (https://www.arduino.cc/reference/en/) to help you out. Since your numeric display only displays 0-9, you can simply display 0 for any value over 9½ seconds.