



WPI



FUNDESTEAM

Construendo • Pensando • Crescendo

Developing a Robotics Curriculum in Panama with Fundesteam

Nathan Johansen

Matthew Boros

Cem Alemdar

Instructors: Professor Chiarelli, Alex Sphar

October 11th, 2018

Developing a Robotics Curriculum in Panama with Fundesteam

An effort led by the Fundesteam corporation to create a robotics curriculum and help improve
STEM education in Panamanian public schools.

An Interactive Qualifying Project Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science
on October 12, 2018.

Submitted By:
Cem Alemdar
Matthew Boros
Nathan Johansen

Submitted to:
WPI Advisors: James Chiarelli, Alex Sphar
Sponsor: Fundesteam



This report represents the work of four WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review

Abstract

STEM (Science, Technology, Engineering, Math) education has been an exponentially growing field all over the world. Panama has been slow in following this trend compared to the rest of the world. Since 2013 there have been several non-governmental organizations whose mission is to teach children in Panama the importance of STEM education, and improve the quality of STEM education in Panamanian schools. One such non-governmental organization is Fundesteam, our sponsor, which is planning on creating a curriculum and proposing it to schools throughout Panama to encourage students between the ages fourteen and eighteen to pursue higher education in STEM fields. They have asked us to create workshops using Arduino technology to teach concepts in physics, robotics, computer science, mechanical engineering, and math.

Table of Contents

Abstract	i
Table of Contents	ii
List of Figures	iv
Executive Summary	v
Authorship	viii
Acknowledgments	ix
2 Background	2
2.1 The Importance of Robotics	2
2.2 STEM Education in Panama	3
2.3 Prerequisites for Robotics Education	5
2.4 Arduino Technology	6
2.5 Developing a Curriculum	6
2.6 Fundesteam’s Solution	9
2.7 Summary	9
3 Methodology	10
3.1 Identifying Obstacles with Teaching Robotics in Panama	10
3.2 Communicating with Primary Stakeholders	11
3.3 The Robotics Curriculum	12
4 Findings/Analysis	15
Workshop Examples	18
Example 1: Introduction to Matrices	18
Example 2: Rocket Launch Simulation	20
Example 3: Ultrasonic Detector	23
5 Conclusions and Recommendations	27
6 References	31
7 Appendices	33
Appendix A	33
Appendix B	47
Appendix C	61
Appendix D	73
Appendix E	81

Appendix F	88
Appendix G	97
Appendix H	105
Appendix I	114
Appendix J	123
Appendix K	138
Appendix L	149
Appendix M	163
Appendix N	172
Appendix O	185

List of Figures

1	An example Arduino circuit	15
2	List of parts included in Arduino kit	17
3	A 4x4 matrix keyboard	18
4	Example Arduino schematic	19
5	A screenshot of the workshop	20
6	Circuit diagram for the rocket workshop	21
7	Graph of distance traveled versus rocket angle	22
8	Ultrasonic detector visual	23
9	Distance, velocity, and acceleration graphs	24
10	Circuit diagram for workshop	25
11	Screenshot of workshop data	26
12	Arduino starter kit	29
13	Arduino circuit diagram	29

Executive Summary

Panama, while better off than many developing nations, is still progressing but is behind many modern standards. As a developing nation, Panama's economy is not the most efficient nor productive. One of Panama's downfalls is its education system. Panama's education system is roughly 5 years behind the USA. A significant way to improve a country's economy is to increase the magnitude of the educated and skilled workforce. This is the goal of the nonprofit and nongovernmental company, Fundesteam, our sponsor.

STEM refers to the technical subjects, which are what Panama's education lacks the most. STEM has been rapidly growing in today's society, both in schools and the workplace. People with STEM degrees experience very high employment rates and vast opportunities. With STEM growing at faster and faster rates around the world, it is imperative that Panama adopts STEM into their education system to quickly advance and not fall further behind.

While Fundesteam has their own classroom, and provides private after-school programs to teach and inspire STEM related subjects, they aim to further their influence on a more national scale. The task they set for themselves is to provide an online robotics curriculum that Panamanian public schools can access and then teach their own students. Before our team's arrival, Fundesteam has created 24 workshops; our task was to continue their effort by making their curriculum larger and more comprehensive.

This curriculum utilizes Arduino technology, an open sourced module which reads computer programming and then controls robotic components to perform certain tasks. It is a great tool for education as it provides an intersection to theoretical learning with hands-on practice.

This project's goal was to aid Fundesteam in improving Panama's education system by implementing an Arduino robotics curriculum in public schools. In pursuit of this goal we sought to accomplish three main research objectives. The first step was to identify potential obstacles

with teaching robotics in Panama. Next was to collaborate with primary stakeholders to obtain as many resources as possible. The final, and main focus of this project was to develop robotics workshops for public school students. We completed these objectives through cooperation with the employees of Fundesteam along with research of Arduino technology and its applications.

Our team worked on designing robotics workshops that were efficient in combining real world problems, aspects that interest students, and academic variety. In our process, we found key attributes of a successful lesson. We made the lessons concentrate on the different subjects of programming, engineering, or math. We varied our lessons through these three topics. These topics all come together to cover the major building blocks of robotics. It is more effective to work on one skill at a time rather trying to master several things at once.

Our next finding is to place our modules in a context with real life problems and examples. This is important because this context reaffirms the student that what they are learning is important and useful. If the student is bored while completing the workshop, they will not absorb the material as well as they should. In addition, seeing the potential impact they could have in society will serve as an inspiration to pursue higher STEM education.

Our third finding is how valuable it is to endorse creativity from the student. This was done by including a challenge at the end of the workshop. This challenge served as a self test for the student to prove to themselves that they genuinely understand the material by utilizing knowledge in the absence of step-by-step instruction. In addition, fostering creativity instills in the student soft skills such as problem solving and critical thinking, which are important and applicable in everyday life.

Our final finding is that it is necessary to utilize Arduino parts that the students have access to. Arduino components come in pre-assembled kits. Therefore, it is key to create workshops with parts to which the students will have access. Otherwise, the lesson and its goal will essentially be lost.

At the conclusion of our work, we have come up with recommendations for future projects. Our first recommendation is to have more communication with public schools. To increase the the effectiveness of the curriculum, future projects should be focused on more direct communication with the audience, rather than relying Fundesteam for guidance. Our second is to expand this project elsewhere. Using our findings, creating a curriculum can be done anywhere that improvement is needed or desired.

Authorship

Abstract:	All
Executive Summary:	Nathan Johansen
1 Introduction:	Matthew Boros
2 Background:	
2.1 The Importance of Robotics:	Cem Alemdar
2.2 STEM Education in Panama:	Matthew Boros
2.3 Prerequisites for Robotics Education:	Nathan Johansen
2.4 Arduino Technology:	Cem Alemdar
2.5 Developing a Curriculum:	Matthew Boros
2.6 Fundesteam's Solution:	Nathan Johansen
2.7 Summary:	Matthew Boros
3 Methodology:	
3.1 Identifying Obstacles with Teaching Robotics in Panama:	Nathan Johansen
3.2 Communicating with Primary Stakeholders:	Cem Alemdar
3.3 Developing a Robotics Curriculum:	Matthew Boros
4 Findings/Analysis:	All
6 Conclusions and Recommendations:	Cem Alemdar and Matthew Boros
5 References:	All
7 Appendix	All

Acknowledgments

The Fundesteam employees for hosting our project.

Marvin Castillo, Lutfi Garzon, and Pedro Pascual

IQP Advisors

Jim Chiarelli and Alex Sphar

1 Introduction

In 2011, an article on the Reuters news website called Panama's education system "the worst in the world". The article stated that the education in Panama was so bad that the country's future could be in grave danger, and the author even hinted that the economic boom seen in Panama at the time could be temporary. The article was correct. In 2011, Panama's GDP had grown 11.8% (WorldBank, 2011). Five years later it only grew 4.9%. This theory is not put forth by just one author. The Central Intelligence Agency's Factbook on Panama mentions how improving the quality of education could drastically improve conditions throughout the country (CIA.gov, 2018).

Education in Panama should be considered one of the most important aspects of the country if its economy is to become successful (Education Counts, 2010). In the past decades, technology has become one of the most important industries in the world, since the way people live has changed monumentally due to technological advances. An improved education system in the technology sector could help Panamanian schools move forward, and open up the world of technology to students (Green & Write, 2015). In mid-2016 Panama's president announced a new website to help educate Panamanians and bring more technology to the country (Presidencia.gob.pa, 2016). More recently, the Fundesteam organization has taken the challenge of developing a robotics curriculum for schools in Panama. The goal of this project is to create a new interactive-based curriculum using Arduino technology to teach students about robotics and its uses in the real world. A widely adopted curriculum could inspire thousands of students to innovate and possibly even start companies based on technology, which in the end could be very beneficial to Panama.

2 Background

Improving Panama's system of education could help the entire country's economy and development. Robotics plays a major role in many of the latest technologies today, so our group focused on a robotics curriculum. In this chapter we will discuss why robotics is important, the amount of education a Panamanian student will have that is relevant to robotics, and how we will implement Fundesteam's solution.

2.1 The Importance of Robotics

The definition of a robot is "a programmable mechanical device that can perform tasks and interact with its environment, without the aid of human interaction" (Larochelle D, 2015). It is one of the fastest growing industries. We live in a world where robots are helping us in kitchens, factories, and even on other planets.

The world and its economies are more technologically advanced than ever before, and the continued statistical growth of the technology industry is inevitable (CompTIA, 2018). In the future, computers are likely going to be the muscle and brain power of work. Robots are also used in many different areas such as surgery, space exploration, deep sea exploration, cooking, etc. Though the debate on robots versus humans is ongoing, there is no denying that robots have helped us in ways we could have never imagined a decade ago. Teaching the younger generation robotics would broaden horizons and expand vision on future directions.

Robotics engineering is a branch of engineering that utilizes knowledge from mechanical engineering, computer science, and electrical engineering. It draws on mechanical functions such as kinematics, stress analysis, and hardware design and uses them to create the physical robot. It takes programming concepts, loops, and algorithms from computer science and uses them to program the robot to perform specific actions and tasks, following a set of specific instructions (code). Electrical engineering concepts such as circuit modelling, signal processing,

and magnetism are employed in designing a functioning robot, which act as a glue between the hardware and the software, ensuring that all of the hardware is able to communicate and perform all of the tasks for which the robot is programmed.

According to Marvin Castillo, founder of Fundesteam, Panama is about five years behind the United States education system. He has also stated that it is even further behind in STEM education, including robotics. In conclusion robotics education is very important for the younger generation to have a better understanding of the current technological advances, and to provide a creative environment for them to create something useful. While learning how to create robots, they will also learn the basic ideas of different engineering concentrations.

2.2 STEM Education in Panama

Science, technology, engineering, and mathematics (STEM) are key components of any robotics education, since robotics typically consists of mechanical engineering, electrical engineering, and computer science. Each of these fields is classifiable as either engineering, or technology. As an example, on WPI's tracking sheet for robotics engineering, seven mathematics courses are required, along with six science courses, three computer science courses, and a total of seven engineering courses (WPI Tracking Sheet, 2018). It is clear that STEM is an integral part of robotics.

Our sponsor Fundesteam also includes arts as a part of the STEM education, rechristening it STEAM. This is an interesting point because it indicates the desire to include the ideas of aesthetic design and creativity into the more traditional STEM education. Our team was asked to keep this under consideration while developing the curriculum, due to the state of the Panama education system.

The Panama education system has been stagnant for decades (Scholaro, 2018). According to Scholaro.com, a website dedicated to providing information about schooling systems, Panama's education system is one of the worst education systems in the world, and

“remains unchanged after 30 years because the government fears upsetting unionized teachers”. The country is faced with a stalemate between the government and a teachers union, and no changes have been made to improve the curriculum. Technology is a very current topic which is unlikely to be included in Panama’s education curriculum. Even if it were included, it would be thirty years out of date due to the stagnant education system, and the technology component of the curriculum would be useless. The substandard and overcrowded public education in Panama pushes many parents to place their children in private schools. The private schools in Panama offer “quality educations relatable to that of North America” (Panama Equity, 2017). Most schools in the United States recognize the importance of STEM education, and have implemented major curriculum changes to include technology and engineering (Reed, 2018). The same has probably been done with private schools in Panama, since they are closely connected with the curricula in North America.

Not every family can afford to send their children a Panamanian private school. The cost of private schooling can be up to twelve thousand dollars per child (Panama Offshore Services). The average monthly wage in Panama was 1238 Balboas in 2016 (TradingEconomics, 2018). This works out to be about \$15,000 per year. The cost of an expensive private school is nearly the Panamanian average yearly wage, which indicates that only a minority of families in Panama can afford to send their children to private schools. Given these inequalities, attempts have recently been made to improve the public education system and include STEM courses. For example, in 2015, about one hundred professors from various schools in Panama attempted to create communities where students could learn about STEM topics and practice what they learned (Laspau, 2015). The attempt was made in a mutual agreement with the professors that there is a need for more developed STEM education in Panama. Clearly the STEM education system is underdeveloped in public schools.

In summary, Panama’s education for public and private schools should be evaluated separately. Students in private schools receive an education similar to that of schools in the

United States, which include all the important topics that are required to move on to any STEM field such as basic mathematics, science, and technology education. The public schools have been stagnant for several decades, and often lack a proper curriculum for technology. A robotics curriculum will boost students' interest in technology and STEM fields, and open up a the world of engineering, science, mathematics, and computer programming to the public schools. Even a basic curriculum that teaches the fundamentals of robotics could spark enough interest to make teachers develop a more advanced curriculum, or possibly inspire students to investigate robotics in their free time.

2.3 Prerequisites for Robotics Education

Robotics is technical field and so some specific knowledge is needed for getting started in robotics. However, we are creating an introductory curriculum to inspire students to pursue STEAM education down the road. So, these students will not be needing knowledge that is very advanced. For high school students, a strong understanding of algebra and geometry and a good foundation of physics will suffice. One of our initial steps should include incorporating an algebra review and a review of the most important concepts of physics for robotics, such as mechanics and basic circuitry. While knowledge in programming would be ideal, it is not necessary as our curriculum will teach what is needed, using simple software and hardware that is very teachable. In Panama, the equivalent of US high school has students choose between two paths for their schooling. They either choose the academic route, or the vocational route (which is focused on technical job training) (Harris S. 2007). This makes our pool of prospective students a little more complex. Considering that the goal for our robotics curriculum is to inspire STEAM interest in students to pursue later in college, it makes sense to have our curriculum focused on the students that choose the academic route.

2.4 Arduino Technology

Arduino is a company that manufactures compact, programmable, modules and provides an open source platform for anyone to use. This platform consist of an interactive development environment and many libraries that have been written and shared by users. Arduino technology has many uses, from gathering and storing data from sensors to making robots. It is easy to learn and simple to use, while being able to perform complex computations. Everything is well documented and there is a huge community of people working with, developing, and sharing Arduino projects around the world and across the internet. Small modules are inexpensive, and well developed, while being limited with memory and number of connection pins. More expensive ones are a lot more powerful and larger. The smaller ones are great for learning the technology and creating projects on one's own.

We aim to use this platform and technology to teach students STEM concepts such as math, robotics, physics, and programming. We hope to show students what they can do with the technology by developing modules and workshops around Arduino. We will approach essential abstract ideas in STEM by creating a physical representation of it. For example, we can use Arduino to teach statistics or electricity. By using mechanical parts like motors and sensors, students will be able to create a simple robot very easily, which will help to encourage them to learn more about how it all works.

2.5 Developing a Curriculum

It is vital that the curriculum our group develops can work for every student, and works around the difficult internal issues of the Panamanian education system. Students who are interested in the robotics program might not have the same background in STEM topics that students in the United States would typically have. Our curriculum has to be designed to work for students with any background. It is possible that thousands of students will experience our

curriculum in the coming years, so it is vital that we optimize the experience as much as possible.

The course our group creates should somehow relate to the outside world, and the experiences a student might have in the future. According to the Flinders University (of Adelaide, Australia) guide on creating a curriculum, the experiences received in a learning environment should “contribute towards their personal, academic and professional learning” (Flinders University, 2018). Our work should make it clear that the robotics skills we are introducing are not useless. In fact, according to Learn.org, robotics engineering job openings are expected to grow by five percent per year between 2014 and 2024, with a median annual salary of around \$80,000 as of 2015 (Learn.org, 2015). In order to spark students’ interest and make the robotics topics we are teaching relevant to their future, we should make it clear that a robotics engineer has real potential to find a stable job that earns a high-level salary. We could also provide famous examples of what robotics engineers have done, such as sending robots to Mars to explore the planet (NASA, 2017). Capturing a students’ attention is one of the most important aspects of a curriculum, because if the student is not engaged with the information being taught, knowledge will not be retained.

According to FAO.org, the Food and Agriculture Organization of the United Nations, there are six key steps to developing a curriculum (El Sawi G. Ph.D, 1996). They are summarized below in the context of our group’s project.

1. **Identify the problem.** What are Panamanian students missing in their public school curriculum? Fundesteam believes that robotics education is missing from the current public school systems.
2. **Evaluate the characteristics of the students.** What do the students know, and what don’t they know? Is there a knowledge gap between what Panamanian students know and the minimum requirements for our curriculum? Our group has

researched what the public schools in Panama teach for STEM topics, but without actually going to schools and talking to students it is impossible to know.

3. **Decide what the objectives are.** What should the students know by the end of the curriculum? Our group along with the Fundesteam employees have to decide what is a reasonable expectation for what students could learn in a 40-week period with meetings occurring once per week.
4. **Choose the relevant content.** This includes textbooks, online materials, and even the materials used in the curriculum. Fundesteam has chosen the Arduino technology as the center of the curriculum, but without further knowledge of the resources available to schools in Panama and the objectives Fundesteam has in mind, we cannot yet decide on the relevant content.
5. **Decide on the methods to achieve the objectives.** How will we teach students the robotics technologies? Will we include visuals, a textbook, hands-on materials, examples of fully-functional robots? These methods should be diversified so we have a technique that works for every student.
6. **Evaluate the effectiveness of the teaching methods.** In Panama we will see how students respond to our methods of teaching. Do they learn better with one method as opposed to another? Do the students find reading a textbook more useful than examining a robot? We should use the teaching methods that work best with the students.

By following these steps, our group will ideally create a curriculum that is effective for every student, and has reasonable goals for what students can learn in the 40-week period. We will test different methods of teaching robotics to students, and keep track of what they enjoy the most, and also what methods are the most effective. As long as we keep in mind what the students do and do not already know, we will be able to create a curriculum that students in

Panama's public schools will be able to easily understand, and learn the fundamentals of robotics.

2.6 Fundesteam's Solution

Fundesteam is our sponsor, and the leader in promoting STEAM education in Panama. This organization is a non-profit and non-governmental. They work on equipping schools with robotics laboratories and curricula, and they organize international robotics competitions, most notably the World Robotics Olympiad. Their goal is to break the cycle of poverty in Panama through educational advancements. They have raised a little more than \$200,000 in donations to put towards their work.

Fundesteam has been able to implement robotics in private schools. They have established after-school sessions to teach the students. However they have struggled to achieve this in the public school realm as the public schools lack the resources of private schools, imposing a significant obstacle for Fundesteam (Pascual, P. 2018).

2.7 Summary

Fundesteam's solution of implementing a curriculum that uses forty weekly meetings to teach robotics is perfect for Panamanian students. It gives students time to experiment with robotics on their own for a week, and then regroup with other students to compare the work they have done and take another step forward with the curriculum. Some issues may arise if the curriculum has too much information to be imparted to the students in a single meeting, such as the basics of programming and electrical and mechanical robotics. Another problem that could occur might be that the Panamanian students' do not have the background to complete the robotics curriculum. Our group will have to design and test the curriculum we create with these issues in mind. Our aim is to provide a way for students to move through the curriculum at their own pace.

3 Methodology

Creating a robotics curriculum for Panamanian students is a huge task. It will require resources from a variety of places such as professional curriculum developers and experts in robotics engineering. Any work we establish in the school systems should be looked over by STEM curriculum development specialists. The final product will take years to perfect. In order to manage our work properly, our team has established three goals.

1. **Identifying Obstacles with Teaching Robotics in Panama.** Identifying obstacles before creating a curriculum will help us make an effective curriculum specifically for Panama. We can also determine what STEM knowledge the students have.
2. **Communicate with the primary stakeholders.** Fundesteam and other invested parties will have a variety of resources for us to reference. They could also have ideas for the curriculum, and more experience with Panamanian schools than our group will have. They will be a great resource for us.
3. **Develop a curriculum.** With Fundesteam we will develop and test a curriculum based on weekly modules specifically optimized for Panamanian students. We will test many different methods of teaching and see which are most effective.

3.1 Identifying Obstacles with Teaching Robotics in Panama

Pre-Project

Implementing a robotics curriculum is not a simple process. There are cultural and economic complexities that need to be taken into account, especially in a developing nation such as Panama. There is data collection that needs to occur first before we begin designing a curriculum. Through communication with Fundesteam, and trips to local public schools, we need to answer the following:

1. What technical and STEM related knowledge do the students already have?

2. How well do they know English?
3. What resources do the schools have for laboratories?
4. Do the schools have teachers with the will and the time to take on teaching a robotics course?
5. What robotics knowledge do the teachers already have and what would we have to teach them?
6. How many students will be interested in pursuing robotics and taking our course?

Our first goal in Panama is to find the answers to these questions because a lot of work on the curriculum will be based on these findings. This information will allow us to determine an approximate student/teacher ratio, how much technology the students will have to share, and what lessons will need to be included in our course. Collecting data will consist of interviews with public school teachers, as well with members of Fundesteam. We also want to reach out to the private schools where robotics education has been a success for advice.

Once all this data is collected it will allow us to create a robotics curriculum that can be successful in the context in which it will be taught. This initial data collection is key to success: creating a robotics curriculum is useless if it is ultimately ineffective where it is being taught.

Post-Project

It is important to review the content and its effectiveness once completed before continuing. This ensures staying on task and continuing to create good content. Every week we reviewed our completed work to make sure it was up to par. This review spurred us to do better work every week.

3.2 Communicating with Primary Stakeholders

It is very important to identify the primary stakeholders before starting to work on collecting data. For this project our primary stakeholders are our sponsor, the students who are going to learn through this curriculum, and the teachers who are going to teach this curriculum.

It is important to identify them because what we do in this project will directly affect these people and organizations. We acknowledge that in our communications with these stakeholders, it is of vital importance that we avoid digressing from our scope and that we strive to understand their expectations of our team and project. We will be using many different ways and media to communicate, such as in-person discussions, emails, video conferences, and even focus groups.

To conclude, we believe communication is key to understanding what the stakeholders expect from us, what we aim to give them, on what points do we agree and disagree, and most importantly, not to misunderstand each other.

Post-Project

We found that Fundesteam has very close communications with the nearby public schools. Many Fundesteam employees spend lots of time at the public schools, teaching workshops and helping students learn. Most employees at Fundesteam speak English fluently, however the students at public schools speak only a basic level.

Since the Fundesteam workers visit schools on a daily basis, we are able to communicate our ideas and goals with all primary stakeholders through Fundesteam. Fundesteam's employees understand the school system very well. They know what knowledge students will have in each grade level, which assisted our curriculum development process. The Fundesteam employees also understand how to effectively teach a robotics lesson to students. By interfacing with Fundesteam, our team could effectively communicate with all primary stakeholders in the curriculum development process.

3.3 The Robotics Curriculum

After assessing the robotics education in Panama and communicating with Fundesteam and other primary stakeholders, our group will be ready to develop and test a curriculum. We will determine what knowledge the students have about robotics, and from there we can decide

on a starting point for the curriculum. We can remove modules from the model curriculum if we believe they are too basic for our audience. After determining a path, students should take from the starting point to the end goal Fundesteam specifies for us, we will begin developing modules, and test them in the school environment in Panama. If we have school permission, we ask a sample of students from different backgrounds to work through the modules, and ask them questions afterwards about the difficulty of the module and how much they learned while working through the problem. We will implement different versions of modules using images, verbal and written instructions, and physical examples of the final product. Most importantly we will place these modules in real life contexts, such as traffic, sustainability, and data collection. This will help students learn the “real life” cases where this information is being used. By testing the different versions of the same module, we can determine which learning methods are most effective, and choose one or more versions to merge into the final product.

By testing each version of our modules on students, we will learn how to optimize the learning experience for Panamanian students. We cannot expect the optimal experience for a student from the United States to be the same as a student from Panama, since the educational background each student has would be vastly different. After testing just a few versions of modules, we can find the optimal learning experience for the average Panamanian student, and follow that pattern for the rest of the curriculum.

To make this curriculum applicable to a larger audience, our group will implement modules that are meant to be prerequisites for robotics education. For example, if one student in Panama has no knowledge of how electricity works and is struggling with the modules in our curriculum, we will offer a module on the basics of electricity necessary to understand our robotics modules. These prerequisite modules will allow students with different levels of knowledge about STEAM to still experience our modules.

By ensuring that our curriculum is applicable to the largest possible audience in Panama, and testing various learning methods on students, we will have developed a

curriculum using forty weekly modules to get students an idea of what the world of robotics is like. Before implementing the curriculum in schools, we will have to complete considerable amount of testing and make sure the modules are in order so it is easy to move from one module to the next. Using these methods we will create an optimal robotics curriculum for Panamanian students.

Post-Project

After talking with our sponsor and teachers at Fundesteam, we have concluded that having modules that focus on different aspects of robotics will address students with different interests and skills. We divided these core aspects into three: Programming, physics and mechanical, and math. Of course each workshop will use all three, but each workshop puts a higher importance on one, and goes into more depth under that subject. This will let the students choose which one they enjoy more and want to learn more in depth.

While deciding on the level of difficulty for each workshop, we decided on putting a section at the beginning that lists the prior knowledge required for that workshop, and which concepts they will learn from it. This helps the students judge for themselves if they will be able to do it. Listing these are important for the teachers as well, because they can use which concepts the students are learning from these workshops. We also use some workshops as stepping stones to tackle more complicated concepts. So, it is easy to see a concept on the “What you will learn” part in one workshop and “What you need to know” part in another.

4 Findings/Analysis

Our team analyzed the most efficient ways to encourage student interest in our workshops. With the help of Marvin Castillo from Fundesteam, we were able to design a workshop template that effectively keeps students interested, teaches new material, and endorses creativity.

Finding 1: Having modules concentrate on programming, mechanical, or math.

By having each module focus on a specific topic in STEM, we allow students to hone in on a specific skill and improve that skill. Some previous workshops had multiple subject areas involved, but those workshops were found to be less effective since students were overwhelmed by the large amount of material being presented. Just like in a classroom, it is much easier to learn one thing at a time than it is to learn several things at once. Our team created workshops in the computer science, mathematics, and mechanical engineering fields, with each workshop trying to improve only one skill at a time. Although many skills can be used in a workshop, most of these skills should be previous knowledge that the students already have, and only one should be a new skill. Our team found that choosing a specific topic in a specific area of STEM leads to more successful results.

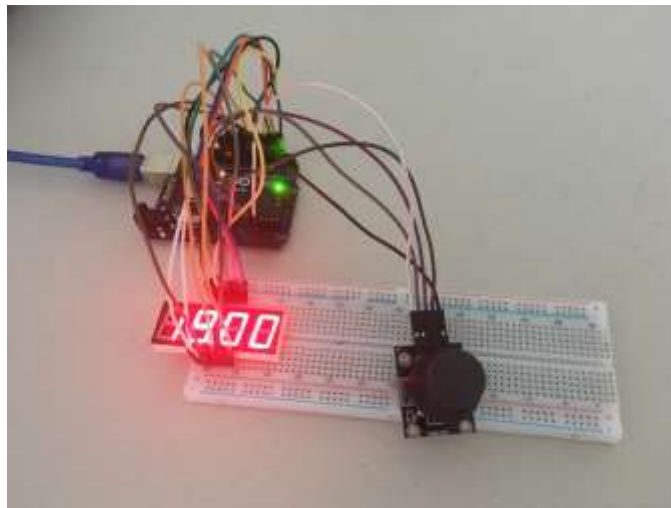


Figure 1 - An example Arduino circuit

Finding 2: Placing the modules in context with real life problems and examples.

Our team found that applying a context to each workshop is the best way to keep students interested in the material. For example, if a workshop involves controlling LED lights on a timer, that workshop could be related to programming a traffic light. By using real-life problems and applying context, the students find workshops more relatable, and therefore more interesting. Workshops without context are simply exercises that to students, do not have a meaning.

Students learn more when they have a consistent interest in the workshop. Without their interest, they can simply do the workshop without paying attention to the details, and learn much less. A context prevents this from occurring by making the workshop relatable.

Finding 3: Endorsing creativity in workshops by including challenges.

Before we started writing these workshops, we spent a few weeks reading, studying, and recreating Arduino workshops made by other people. These included many we found on the internet and some that our sponsor has provided us. They also asked us to use the ones they provided as a template for the workshops we were going to create. Even though the workshops they gave us were all in Spanish, we managed to understand the structure enough to recreate them.

One aspect that stood out in the Spanish workshops were the inclusion of challenges at the end of each workshop. These challenges would use the concepts they taught in the workshop, and let the students come up with their own way of doing something similar to which they created in the workshop. We realized that giving a certain creative freedom to the students will encourage them to come up with their own unique approach to the challenge.

Also considering that many students will be working on these workshops in a classroom environment, having the same product as everyone else at the end would not be rewarding

enough. So the challenges let them create an end product that they can call theirs, and not a product that someone else designed for them to recreate.

Finding 4: Using Arduino parts that all students have access to.

When we are considering implementing a curriculum it is important to know what resources we, the students, and the teachers have. Fundesteam has supplied us with three different Arduino kits, which were composed of the same, or similar mechanical parts. We were told that the students will be given the “starter” kit, which was one of the three that we had. We had to make sure before starting a workshop that all the parts we were using were also in the starter kit. Even though our sponsor told us that the other parts were easily accessible and inexpensive, we believe that it would be disappointing if the students opened up their kits and started a workshop just to find out that they don’t have all of the required parts for that workshop. The starter kit on which based our workshops contains the following parts:



Figure 2 - List of parts included in Arduino kit

Workshop Examples

Example 1: Introduction to Matrices

This workshop is a basic introduction to matrices, matrix operations, and matrices in programming. It utilizes a matrix keypad, which is a 4x4 matrix of buttons. The workshop starts with an explanation of what a matrix is, where they are used, and how they are helpful. Then it explains that you are going to create a matrix calculator. The good part about the workshop is that the students will be able to create a matrix calculator in a couple hours without not knowing how matrices work at the start of the project.

The workshop then goes in depth explaining the 4x4 matrix keyboard and how it works. How it is connected to the Arduino, and how it is programmed. Then it introduces the mathematical introduction of what matrices are. Briefly explains basic matrix operations, addition and subtraction.

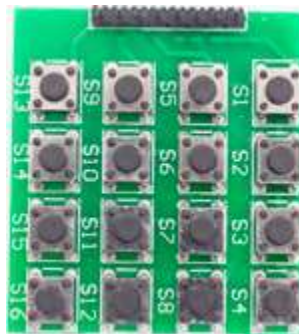


Figure 3 - A 4x4 matrix keyboard

After showing the whole code for this activity, the lesson explains how matrices are used in programming. It explains that an array is a one dimensional matrix, and that an array of arrays are used to create two dimensional matrices. It goes into examples of how to initiate and use matrices in programming, and shows example uses of matrices in the code for the activity. Then it discusses a programming concept called switch cases, and explains what they do, and then gives examples of how to use them. The lesson explains how switch statements were

utilized in the activity code. Finally, students are taught how to creating your own programming functions in the Arduino IDE. They are taught how to use them, how to create them, and how they make code easier to read and understand. All of the examples and references to the activity code are color coded to easily understand what piece of code does what task.

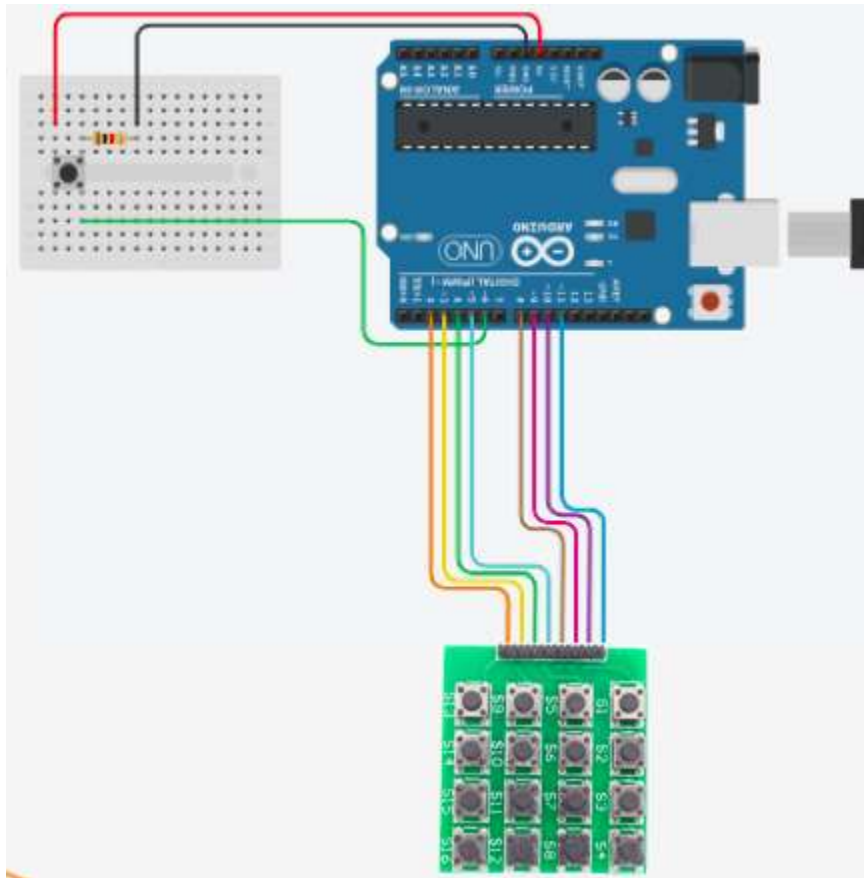


Figure 4 - Example Arduino schematic

After learning all the concepts, trying out the matrix calculator they wired on a breadboard, and copying the code they were given, the students are given a challenge. They are asked to create a small synthesizer/sound board. In order to do this they have to connect a buzzer into the circuit, and change the code so that the buzzer plays a certain frequency when you press certain buttons on the matrix keyboard.

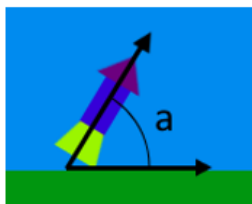
Example 2: Rocket Launch Simulation

The rocket launch simulation workshop places the student in a situation where they have to launch a rocket in an open field, and make the rocket travel the farthest horizontal distance as possible by changing the angle at which it is launched. This workshop first provides students with what prerequisite knowledge they need to know in order to complete the lesson. It also provides a list of concepts that will be taught in the lesson. This allows students to quickly look at the workshop and decide if they can complete it.

The lesson moves into what an angle is, and how they are measured. Arduino lessons tend to use lots of visualization to help students understand what is going on along the way.

F. Activity Description

An angle is a way to measure the space between two lines that meet at a point. For example, in the image shown, the angle is represented using "a".



Angles are measured using degrees. Using the degrees measurement, a full circle is 360 degrees. A half circle is 180 degrees. The following is a diagram to show how degrees work. The symbol for degrees is $^{\circ}$.

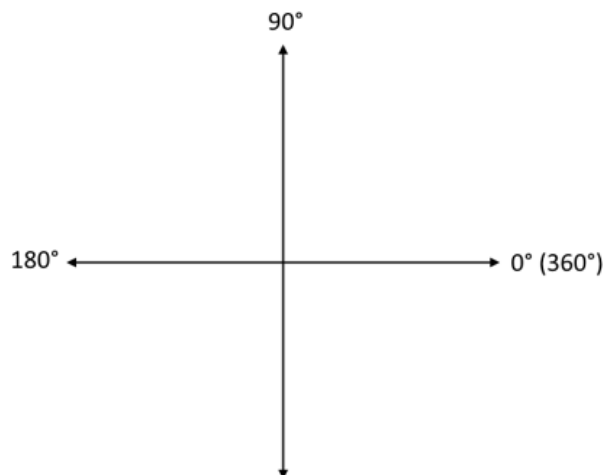


Figure 5 - A screenshot of the workshop

The workshop introduces trigonometry, and explains how to break a vector into its X and Y components using sine and cosine. Examples of sine and cosine are given, and practice problems are provided, along with solutions. The practice problems included in the lesson ensure that the students understand concepts that are taught. It also allows for independent work. A student can complete the problems, look at the solutions, and grade themselves without any outside help.

The lesson then goes into basic kinematics equations. It breaks down the logic of how to calculate the rocket's horizontal distance into smaller problems. The smaller problems are solved, and then those answers are brought back to the original equation to solve the entire exercise. An example of a thirty degree angle is used to show how these equations work.

At this point in the project, the students will understand the physics and math behind the problem. They know how to take a rocket's angle and calculate the horizontal distance traveled. The circuit diagram for the Arduino is provided, and code is included in the workshop. The student creates the circuit, plugs in the Arduino, and compiles and uploads the code.

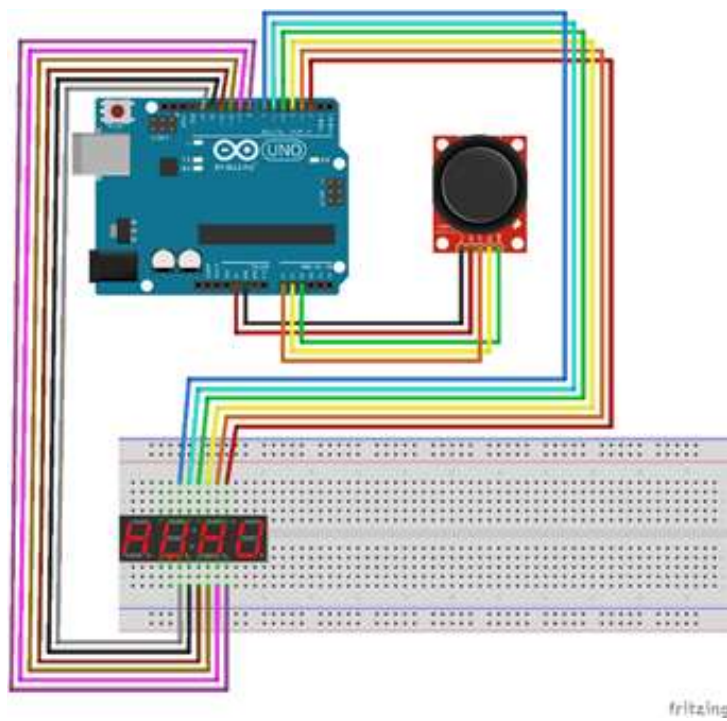


Figure 6 - Circuit diagram for the rocket workshop

The joystick in the circuit allows a student to select the angle that the rocket is launched at. The digital display shows which angle is currently selected. When the joystick is pressed down, a simulation of the rocket launch is shown on the computer screen. After the simulation is completed, the distance traveled by the rocket is shown on the digital display. Students are instructed to collect fifteen pieces of data on rocket launches, and plot the points. On the X axis is the angle that the rocket is launched at. On the Y axis the distance traveled is plotted. Students will then try to estimate which angle gives the maximum distance, using the data they have collected.

After estimating the best angle for distance, the students are given the solution. Forty five degrees gives the maximum distance. They are shown the complete curve that they outlined using the fifteen data points they collected.

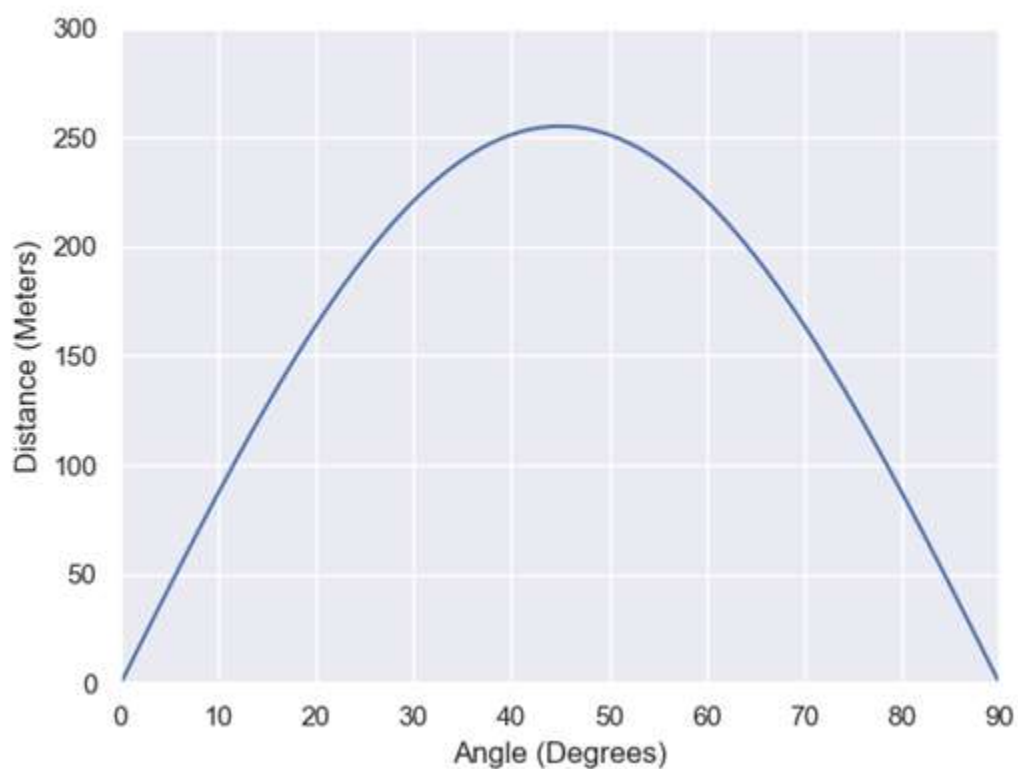


Figure 7 - Graph of distance traveled versus rocket angle

At the end, the students are challenged to complete another task. They are asked what angle would give maximum distance if the rocket were put on a cliff above the flat plane. They are asked to use intuition if the math is too difficult. In the entire workshop, only basic algebra is used. The formulas involved make it easy to solve for the required variables. No quadratic equations are involved, to simplify the learning process for the students.

Example 3: Ultrasonic Detector

The ultrasonic detection workshop teaches the students about kinematics in physics. This workshop teaches the theory behind displacement, velocity, and acceleration; and the relationships between them. Then, it has them test what they learned in an experiment. This workshop also teaches skills of data management, formulation, and graphing in Microsoft Excel.

Most notably, the workshop utilizes the ultrasonic detector and the push button. The workshop starts off by explaining the importance and applications of the parts in the circuit have in the real world. It then transitions into how these parts work. It goes into detail about what they do and how they do it, while providing visuals like the diagram below. After the parts are explained, the workshop teaches how to use and wire them.

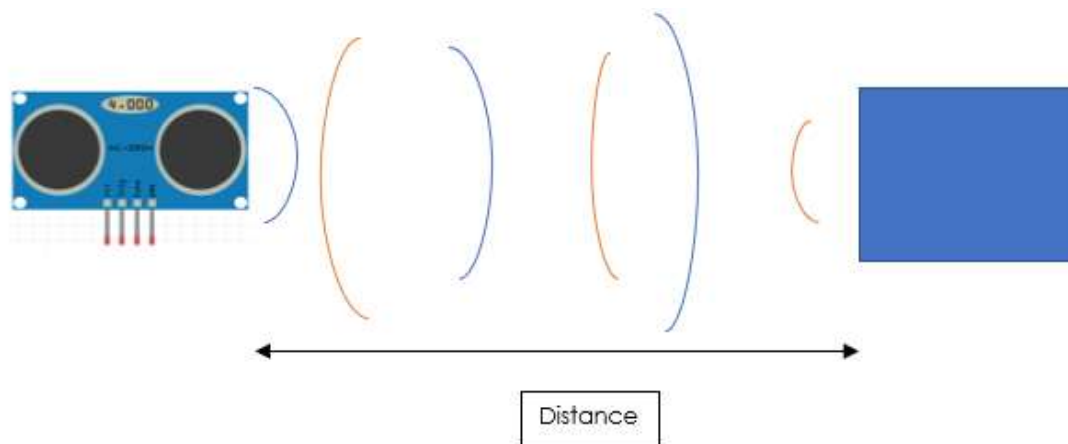


Figure 8 - Ultrasonic detector visual

After explaining the key parts of the circuit, the workshop teaches the math and theory of displacement, velocity, and acceleration. It goes through an explanation of the what each one is and the math behind them while demonstrating example graphs as shown below.

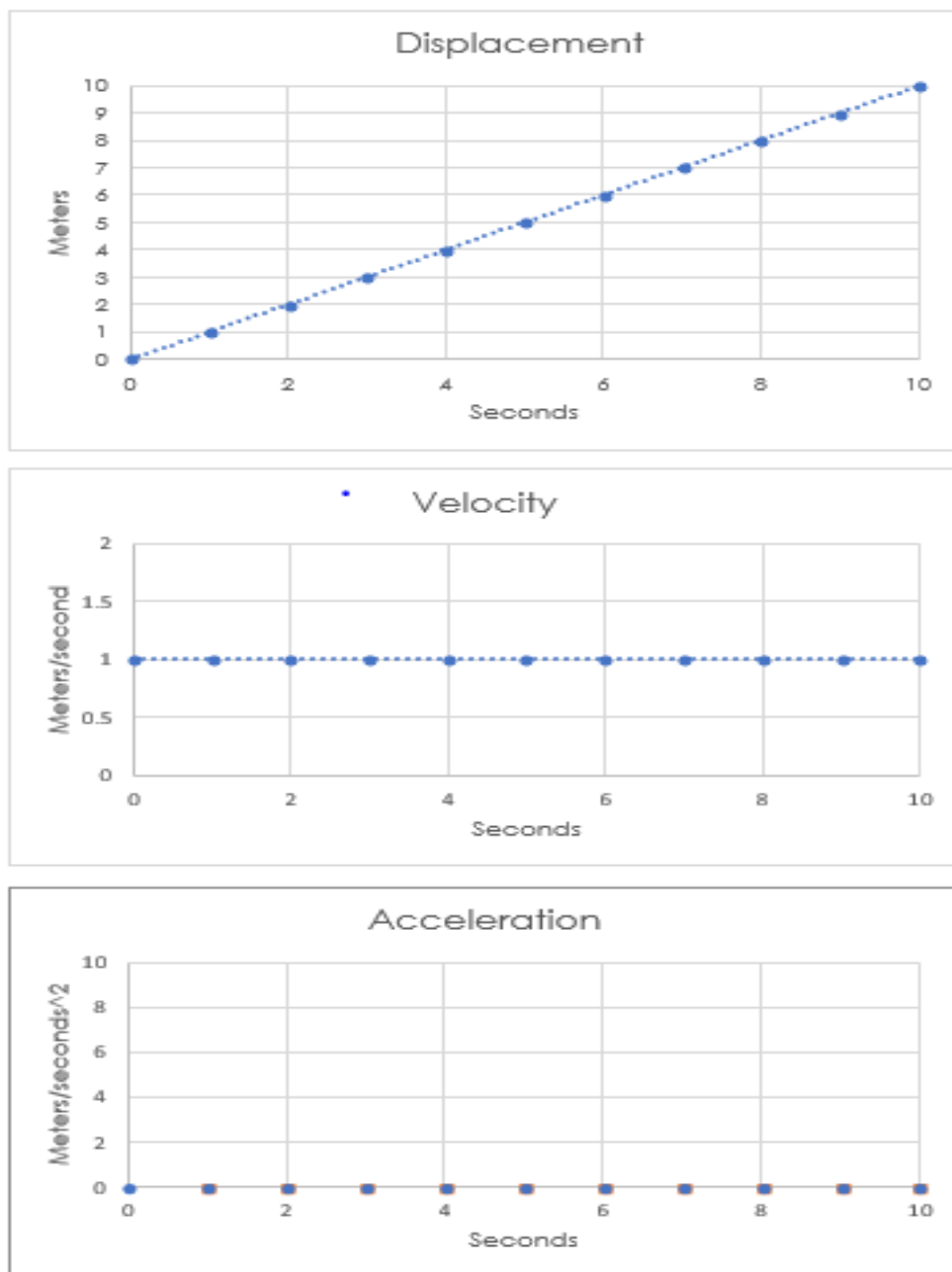


Figure 9 - Distance, velocity, and acceleration graphs

At this point in the workshop, the Arduino parts and the theory of the lesson have been taught. The workshop shifts into having the student set up the circuit and conduct an experiment. The student is to push down on the button which tells the ultrasonic detector to read and record displacement of a two-liter bottle rolling. This gives the student a set of data for zero-acceleration motion. Once this first step is complete, the workshop teaches them how to put this data into Excel. The workshop takes them step-by-step on how to sort the data, and how to make formulas to obtain data for velocity and acceleration. The workshop provides several screenshots so the students know what progress and kind of data they should have. After they have collected and analyzed their data, they are taught how to graph it. Then their experimental graphs should roughly resemble the earlier example graphs shown earlier.

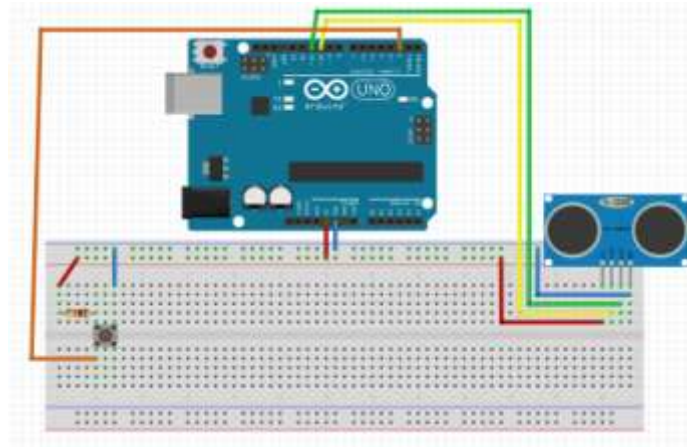


Figure 10 - Circuit diagram for workshop

Time(milliseconds)	t(s)	t(s)(adjusted	D(cm)	D(m)	V	a
3833	3.833	0	7	0.07		
3935	3.935	0.102	9	0.09	0.147059	
4037	4.037	0.204	10	0.1	0.196078	0.245027
4139	4.139	0.306	13	0.13	0.197044	-0.237907
4240	4.24	0.407	14	0.14	0.147783	-0.004758
4342	4.342	0.509	16	0.16	0.196078	0.236741
4444	4.444	0.611	18	0.18	0.196078	0.000000
4546	4.546	0.713	20	0.2	0.196078	-0.004689
4648	4.648	0.815	22	0.22	0.195122	-0.004666
4751	4.751	0.918	24	0.24	0.195122	0.000000
4853	4.853	1.02	26	0.26	0.195122	
4956	4.956	1.123	28	0.28		

Figure 11 - Screenshot of workshop data

Once the students have successfully plotted their data correctly, they begin the challenge portion of the workshop. Here they are they not directly taught something new, but they are supposed to have more freedom and independence to demonstrate what they have learned in the workshop. The challenge of this workshop is for them to repeat this experiment, except doing it with a situation where there is a constant non-zero acceleration. There is a suggestion to record an object in freefall, where gravity will serve as the constant acceleration. However, the student is free to be creative and is allowed to use mind to think of scenarios where there is constant acceleration.

5 Conclusions and Recommendations

With the guidance of Marvin Castillo, the CEO of Fundesteam, we successfully created fifteen Arduino lessons to be implemented in weekly workshops, available to over three hundred schools in Panama. These lessons are available online in English and Spanish. Fundesteam informed us of what Panamanian students know and what they don't know, and helped us create effective workshops.

Objective 1: Identifying Obstacles with Teaching Robotics in Panama.

In our time at Fundesteam, our team learned more and more about the Panama education system, and the knowledge that students have the time by the time they reach high school. We worked with education experts at Fundesteam to ensure that our workshops would be effective in teaching Panamanian students about robotics. At the start of each workshop we have created, we have a section dedicated to prerequisite knowledge to complete the workshop, and also a section for learning goals. These sections in the workshop template will help students make sure they can complete the lesson before starting it, so no time is wasted.

Objective 2: Communicate with the primary stakeholders.

Although we did not directly communicate with the public schools in Panama, Fundesteam employees spend time every week at nearby schools to learn about their needs and also help teach the workshops being created. Since Fundesteam is in constant communication with the public schools, we have communicated with the primary stakeholders by interfacing with Fundesteam.

Objective 3: Develop a curriculum.

Each member of our team created one completed Arduino lesson every week. The lessons were presented to employees at Fundesteam and then placed in a review process before being implemented into the curriculum. With the guidance of Marvin Castillo, we learned how to create effective Arduino lessons for the students. One of the key aspects of the workshops

was the real-world context they had to be applied to. For example, if a workshop was created to control various LEDs, the context could be creating a traffic light for an intersection. Our team learned that if the workshops have an interesting context, the students will be engaged in the lesson, and are more likely to retain the knowledge they learn.

Our overall experience with the project.

When we came in the first day of the project, our sponsor was excited to work with us, and he emphasized the importance of the work we were going to do. We believe his positive attitude and his emphasis on the problems we were trying to solve gave us a motivational boost that helped us concentrate, and create well-researched, content rich workshops. We were given full creative freedom while designing these workshops, which let us come up with many different ideas. We would usually come up with a certain topic we wanted to teach, like statistics or matrices, and build a workshop around that topic, or we would build a workshop around a single mechanical part and try to teach different concepts using that part.

Our work environment was an office. We had air conditioning, table space to work on the Arduino kits while using our laptops to write the workshops. The staff working there was helpful and cheerful. They were also very knowledgeable about the STEAM learning environment. Most of the people we worked closely with were engineers/teachers at the company and were only a few years older than we, so we didn't have a problem fitting in to the work environment. It was also a pet friendly office, so there were few dogs with us which were distracting at times, but usually they kept us entertained while we were sitting in an office all day writing the workshops.

We have definitely learned a lot from this project ranging from using Arduino's to figuring out ways to teach abstract concepts using physical modules. We were given one week to experiment with the Arduino kits to learn how to use them. We were also given example workshops to use as a template while creating our own, but they were also helpful while we were trying to learn how Arduino works. We also learned how a small non-governmental organization runs. We learned to communicate with people who didn't speak English because

we had to. Finally, we believe we learned to push ourselves to learn outside of our comfort zone. Studying for a subject or working on a project at home is very different than going to work every morning and getting work done in an office with your colleagues.



Figure 12 - Arduino starter kit

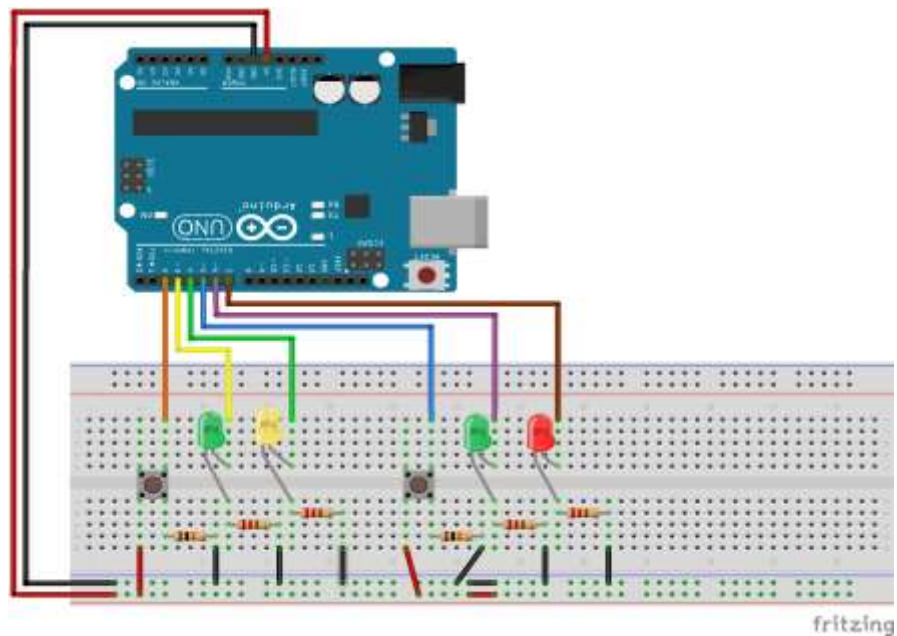


Figure 13 - Arduino circuit diagram

Recommendations for Future Projects

While our project was focused specifically on the Panama education system, the architecture for this workshop-based curriculum could be applied anywhere. Arduinos are inexpensive small modules, which can be used to teach mechanical, electrical, and software engineering concepts. The beginner model called Arduino UNO is about fifteen to twenty dollars, however a whole kit with an Arduino UNO and mechanical parts to connect can increase this price. The kits we used were priced at thirty dollars.

Students can find the lessons they are able to do by looking at the prerequisites section, and choose workshops that interest them. By allowing the students to choose which workshops they want to complete, the curriculum keeps the students engaged and interested, while following a stable learning curve between the different subjects.

This curriculum design is not effective for just Panama. It can be applied anywhere. By translating the current workshops into various languages and continuing to create new workshops, this curriculum design is an inexpensive and effective option for any school system seeking to introduce engineering concepts.

For future projects, we recommend visiting schools at the start of the seven week period to get as much information about the users of the workshops as possible. It is extremely useful to know what skills the students have, and what skills they do not have. It is also useful to find out what kinds of resources the schools have available to them. This will help in the curriculum building process, since it is vital to know your audience before creating the workshops.

6 References

- CIA. (2018, March 26). The World Factbook: PANAMA. Retrieved April 03, 2018, from <https://www.cia.gov/library/publications/the-world-factbook/geos/pm.html>
- CompTIA. IT Industry Outlook 2018. (2018). Retrieved October 7, 2018, from <https://www.comptia.org/resources/it-industry-trends-analysis>
- Davis, K. (n.d.). Evaluating the Difference: Private vs. Public Schools in Panama. Retrieved April 9, 2018, from <https://www.panamaequity.com/blog/evaluating-difference-private-vs-public-schools-panama/>
- Earl, D. (2010). How can tertiary education deliver better value to the economy? *Education Counts*. Retrieved October 7, 2018, from https://www.educationcounts.govt.nz/_data/assets/pdf_file/0004/86980/value-of-tertiary-education.pdf.
- El Sawi, G., Ph.D. (1996). Curriculum Development Guide. Retrieved April 09, 2018, from <http://www.fao.org/docrep/009/ah650e/AH650E00.htm#Contents>
- Flinders University. (2018) A Curriculum Development Process. (n.d.). Retrieved April 9, 2018, from <http://www.flinders.edu.au/teaching/teaching-strategies/curriculum-development/a-curriculum-process.cfm>
- Government of Panama. Panama Advances on the Commitment to Modernize Education and Bring More Technology to the Population. (2016, July 22) Retrieved April 03, 2018, from <https://www.presidencia.gob.pa/en/News/Panama-advances-on-the-commitment-to-modernize-education-and-bring-more-technology-to-the-population>
- Greicius, T. (2017, August 3). Mars Exploration Current Missions. Retrieved April 9, 2018, from https://www.nasa.gov/mission_pages/mars/missions/index.html
- Harris, S. "Panama's Great Challenge: Reforming the Educational System" (2007). A new path for Panama. Paper 3. <http://preserve.lehigh.edu/perspectives-v25/3>
- Internet Live Stats. Panama Internet Users. (2016, July 1). Retrieved April 3, 2018, from <http://www.internetlivestats.com/internet-users/panama/>
- Larochelle, D. (2015). 2.1: What is Robotics? Retrieved May 05, 2018, from <https://curriculum.vexrobotics.com/curriculum/intro-to-robotics/what-is-robotics.html>
- Laspau. First Phase of Panama STEM Education Project Comes to a Successful Close. (2015, October 14). Retrieved April 9, 2018, from <http://www.laspau.harvard.edu/panama-stem-education/>

- Learn.org. (2015) Robotics Engineering Majors: Salary and Career Facts. (n.d.). Retrieved April 9, 2018, from https://learn.org/articles/Robotics_Engineering_Majors_Your_Salary_and_Career_Questions_Answered.html
- Mattson, S., & Teran, A. (2011, September 06). Education trap threatens Panama's economic boom. Retrieved April 03, 2018, from <https://www.reuters.com/article/us-panama-education/education-trap-threatens-panamas-economic-boom-idUSTRE7857D420110906>
- Pascual, P. Fundesteam. (2018). Our Mission TO CHANGE LIVES. Retrieved April 03, 2018, from <https://www.fundesteam.org/#our-vision>
- POLS Attorneys. List of the Best Schools in Panama. (n.d.). Retrieved April 9, 2018, from http://www.panama-offshore-services.com/choosing_the_best_panama_school.htm
- Reed, P. A. (2018). Technology Education Standards in the United States: History and Rationale. Retrieved April 9, 2018, from https://link.springer.com/content/pdf/10.1007/978-3-319-44687-5_9.pdf
- Scholaro. (2018) Education System in Panama. (n.d.). Retrieved April 9, 2018, from <https://www.scholaro.com/ed/countries/Panama/Education-System>
- Sergeyev, A., Alaraje, N., Kuhl, S., Meyer, M., Kinney, M., & Highum, M. (2015). *Innovative Curriculum Model Development in Robotics Education to Meet 21st Century Workforce Needs*(Publication). Retrieved April 3, 2018, from American Society for Engineering
- Trading Economics. Panama Average Monthly Wages. (2018, October 11). Retrieved April 9, 2018, from <https://tradingeconomics.com/panama/wages>
- TutorialsPoint. (2018, February 08). Retrieved April 10, 2018, from https://www.tutorialspoint.com/arduino/arduino_overview.htm , E. (2016, July 22)
- Waddell, J. (2015, March 27). The Role of Technology in the Educational Process. Retrieved October 7, 2018, from <https://edwp.educ.msu.edu/green-and-write/2015/the-role-of-technology-in-the-educational-process/>
- Worcester Polytechnic Institute. ROBOTICS ENGINEERING MAJOR Program Tracking Sheet. (2017). Retrieved April 9, 2018, from [https://www.wpi.edu/sites/default/files/inline-image/Offices/Academic-Advising/Robotics \(2021\).pdf](https://www.wpi.edu/sites/default/files/inline-image/Offices/Academic-Advising/Robotics%20(2021).pdf)
- World Bank. Panama GDP Growth. (n.d.). Retrieved April 9, 2018, from <https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG?end=2016&locations=PA&start=2006>

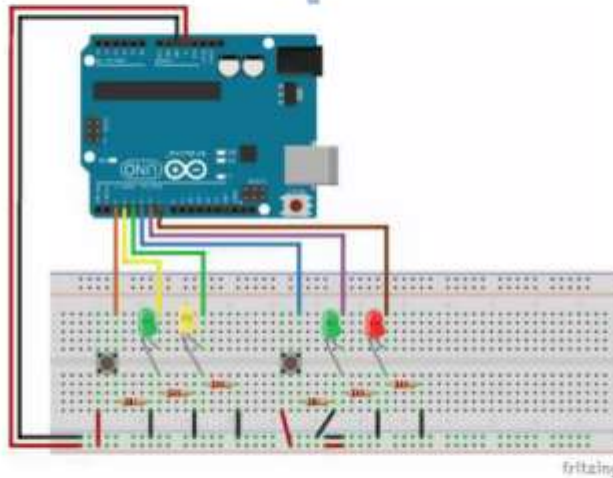
7 Appendices

Appendix A



INTRODUCTION TO PROBABILITY

[Arduino STEAM]





A. Project Title

Introduction to Probability
 (Picking marbles out of a bag)

B. Subjects:

Math, probability.

C. Knowledge

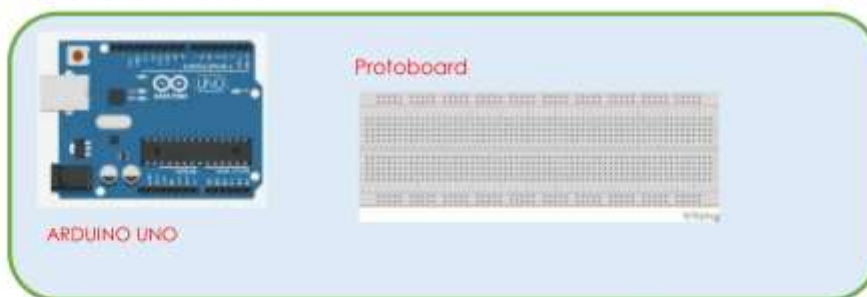
What You Should Know (Prior Knowledge)

- Fractions, percents, decimals, and conversions
- Arduino IDE

What You Will Learn (Future Knowledge)

- Sets and set operations
- Event, outcome, and Venn Diagrams
-

D. Materials Needed



E. Abstract

Probability is used in most STEM fields. It is important to have a basic understanding of probability in everyday life and in a job, as it is a commonly used skill. This course gives an introduction to probability.



F. Context

You are at a nearby rainforest, and someone informs you that there is a 31 percent chance that there is less rain and above average heat in the rainforest this year, which could be bad for the ecosystem. You want to understand what exactly 31 percent means. What does it represent? For this you need probability—an extremely useful part of statistics.

G. Activity Description

In this lesson you will learn the basics of probability, and how to interpret different numbers, such as 31 percent. You will learn what an event is, and the how to read and write probability statements in mathematical notation. You will also learn about experimental probability.

Implementation

The probability of an **event** occurring is a number between 0 and 1 that represents how likely it is that the event will occur. An event can be any sort of occurrence. For example, if the probability of passing a test is 0.75, this means that out of 100 test takes, you are expected to pass 75 of the tests, and fail 25 of them.

Probabilities can also be presented as fractions. For example a 0.75 chance of passing a test is the same as a $\frac{3}{4}$ chance of passing the test.

A probability of 0 means the event will never occur. A probability of 1 means that the event will always occur.

In mathematical notation, an event is typically represented using a letter. For example, we could say

A = the student passes their test

Now whenever we refer to A , it represents the event where the student passes their test. We can represent probabilities easily:

$$P(A) = 0.75$$

The P in this equation means “the probability of”. The entire statement is read as “The probability of A occurring is 0.75”. However, we know what the definition of A is, so we can say “The probability of the student passing his or her test is 0.75”.

Some mathematicians prefer to represent events using words. So an equation could look like this:

$$P(\text{the student passes their test}) = 0.75$$

Any notation is acceptable as long as it is easily understood.

Calculating a Basic Probability



An **outcome** is a possible result in the current scenario. For example, the outcomes of rolling a dice is 1, 2, 3, 4, 5, 6. There are no other outcomes, because the dice does not have any other numbers on it. Another example of an outcome is a letter grade on a test. The outcome of a test can be A, B, C, D, or F. Even though it may seem unlikely to fail a test, it is still listed as an outcome because it is possible.

Let's say there's a probability you want to calculate. The general equation to calculate a probability is:

$$P(A) = \frac{\text{number of outcomes where A is true}}{\text{total number of outcomes}}$$

Example

You have a fair 6-sided dice and you want to know the probability of rolling a 2 or a 3.

G = the dice rolls a 2 or 3

For example, let's list the outcomes of rolling a dice where G is true. You can roll a 2 or a 3, and A will be true, so the numerator in this equation is 2.

Now let's list all possible outcomes of rolling a dice. The dice can roll a 1, 2, 3, 4, 5, or 6. There are 6 possible results, so the denominator is 6.

$$P(G) = \frac{2}{6} = \frac{1}{3}$$

This equation is only true if the probability of each outcome is equal. For now, we will only work with situations where each outcome has an equal probability of occurring.

Event Compliments

Sometimes you want to show the probability of an event not occurring. For example if A is the student passing their test, we want a way to show the student failing their test. We write this as \bar{A} . \bar{A} is called the compliment of event A is pronounced "Not A ".

\bar{A} = A does not occur (the student does not pass their test)

If we know the probability of A occurring, $P(A)$, the probability of A not occurring is calculated by:

$$P(\bar{A}) = 1 - P(A)$$

So if we know $P(A) = 0.75$, then $P(\bar{A}) = 1 - 0.75 = 0.25$.

Practice

B = the result of rolling a 6 sided dice is 3





C = the result of rolling a 6 sided dice is 5 or 6

$$P(B) = \underline{\hspace{2cm}}$$

$$P(C) = \underline{\hspace{2cm}}$$

$$P(\bar{C}) = \underline{\hspace{2cm}}$$

$$P(B) + P(C) = \underline{\hspace{2cm}}$$

Set Operations

Set operations are very useful in calculating probabilities. We will go over a few basic operations in this lesson.

A set is a unique collection of unordered objects. For example $\{1, 2, 3, 4, 5, 6\}$ is a set. $\{1, 2, 2, 3, 4, 5\}$ is not a set because it has "2" twice. $\{A, B, C, D, F\}$ is a set, and represents the possible outcomes of a test. We will use sets to represent outcomes to calculate probabilities.

AND

The AND operation is one of the most fundamental concepts across Mathematics, Computer Science, and any field involving logic. When two sets are AND together, the result is a set that contains all elements that are in both sets. The symbol for AND is \cap . This operation is also called intersection.

Examples

$$\{1,2,3,4\} \cap \{4,5,6,7\} = \{4\}$$

$$\{A, B, C, D, F\} \cap \{A, B, C, D\} = \{A, B, C, D\}$$

$$\{\text{red, orange, yellow}\} \cap \{\text{green, blue}\} = \{\}$$

$$\{5,6,7\} \cap \{7,6,5\} = \{5,6,7\}$$

$$\{\} \cap \{\text{apple, orange, banana}\} = \{\}$$

OR

The OR operation is another fundamental operation in logic. When two sets are OR'd together, the result is a set containing both the elements in the first set and the elements in the second set. The symbol for OR is \cup . OR is sometimes called union.

Examples

$$\{1,2,3,4\} \cup \{4,5,6,7\} = \{1,2,3,4,5,6,7\}$$

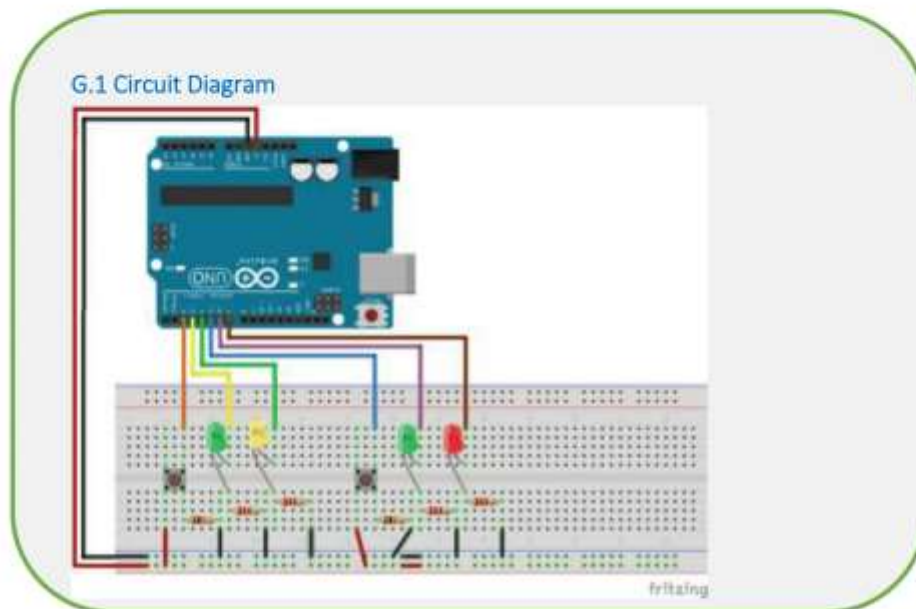
$$\{A, B, C, D, F\} \cup \{A, B, C, D\} = \{A, B, C, D, F\}$$

$$\{\text{red, orange, yellow}\} \cup \{\text{green, blue}\} = \{\text{red, orange, yellow, green, blue}\}$$



$$\{5,6,7\} \cup \{7,6,5\} = \{5,6,7\}$$

$$\{\} \cup \{\text{apple, orange, banana}\} = \{\text{apple, orange, banana}\}$$





G.2 Code

```
1. int BUTTON1 = 2;
2. int LED1 = 3;
3. int LED2 = 4;
4. int BUTTON2 = 5;
5. int LED3 = 6;
6. int LED4 = 7;
7.
8.
9. void setup() {
10.  pinMode(LED1, OUTPUT);
11.  pinMode(LED2, OUTPUT);
12.  pinMode(LED3, OUTPUT);
13.  pinMode(LED4, OUTPUT);
14.
15.  pinMode(BUTTON1, INPUT);
16.  pinMode(BUTTON2, INPUT);
17. }
18.
19. void loop() {
20.  if (digitalRead(BUTTON1)) {
21.    digitalWrite(LED1, LOW);
22.    digitalWrite(LED2, LOW);
23.    delay(200);
24.    int num = random(0, 10);
25.    int r = num < 5;
26.    digitalWrite(LED1, r);
27.    digitalWrite(LED2, !r);
28.    delay(200);
29.  }
30.
31.  if (digitalRead(BUTTON2)) {
32.    digitalWrite(LED3, LOW);
33.    digitalWrite(LED4, LOW);
34.    delay(200);
35.    int num = random(0, 10);
36.    int r = num < 4;
37.    digitalWrite(LED3, r);
38.    digitalWrite(LED4, !r);
39.    delay(200);
40.  }
41. }
```



G. Activity Description (continued)

This circuit represents two bags of marbles. If you press the corresponding button, the LED will display which color marble you have picked.

For this lesson we will use the following event definitions:

$A = \text{marble picked from Bag A is green}$

$B = \text{marble picked from Bag B is green}$

Bag A has 5 green marbles and 5 yellow marbles in it. Bag B has 4 green marbles and 6 red marbles in it. The marbles in each bag have a number from 1 to 10 on them. In Bag A, the marbles with numbers 1 through 5 are green, and in Bag B, the marbles with numbers 1 through 4 are green.

First we'll go over some basic probability questions.

Problems

$$P(A) = \underline{\hspace{2cm}}$$

$$P(B) = \underline{\hspace{2cm}}$$

$$P(A \cap B) = \underline{\hspace{2cm}}$$

Solutions

$$\begin{aligned} P(A) &= \frac{P(\text{marble picked from Bag A is green})}{\text{number of outcomes where A is true}} \\ &= \frac{\text{total number of outcomes}}{1 \leq \text{marble number} \leq 5} \\ &= \frac{\text{number of marbles}}{5 \text{ green marbles}} = \frac{5}{10} = \frac{1}{2} \end{aligned}$$

$$\begin{aligned} P(B) &= \frac{P(\text{marble picked from Bag B is green})}{\text{number of outcomes where B is true}} \\ &= \frac{\text{total number of outcomes}}{1 \leq \text{marble number} \leq 4} \\ &= \frac{\text{number of marbles}}{4 \text{ green marbles}} = \frac{4}{10} = \frac{2}{5} \end{aligned}$$

$$\begin{aligned} P(A \cap B) &= \frac{P(\text{marble from Bag A is green AND marble from Bag B is green})}{\text{number of outcomes where } A \cap B \text{ is true}} \\ &= \frac{\text{total number of outcomes}}{\hspace{10em}} \end{aligned}$$

Let's write out the possibilities for picking out of the bags. Below are the combinations of marbles you can pick where all marbles are green. The numbers on top are from Bag A, and the numbers on the side are from Bag B. The combinations where both marbles are green are highlighted.





	1	2	3	4	5	6	7	8	9	10
1	G,G	G,G	G,G	G,G	G,G	R,G	R,G	R,G	R,G	R,G
2	G,G	G,G	G,G	G,G	G,G	R,G	R,G	R,G	R,G	R,G
3	G,G	G,G	G,G	G,G	G,G	R,G	R,G	R,G	R,G	R,G
4	G,G	G,G	G,G	G,G	G,G	R,G	R,G	R,G	R,G	R,G
5	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R
6	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R
7	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R
8	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R
9	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R
10	G,R	G,R	G,R	G,R	G,R	R,R	R,R	R,R	R,R	R,R

Count how many green boxes there are. It is a rectangle, with height of 4 and width of 5, so there are $4 \times 5 = 20$ boxes. Notice the height of 4 corresponds to 4 marbles in Bag B and 5 marbles corresponds to the width of 5. AND (\cap) operations indicate that you will use multiplication in the problem

Now count how many boxes there are in total. The table is a rectangle with a width of 10 and height of 10, so there are $10 \times 10 = 100$ boxes. This is the total number of outcomes of picking from both bags.

Now we have:

$$P(A \cap B) = \frac{\text{number of outcomes where } A \cap B \text{ is true}}{\text{total number of outcomes}} = \frac{20}{100} = \frac{1}{5}$$

We now know that AND indicates multiplication. If we use this fact, we can calculate this problem by stating:

$$\begin{aligned} P(\text{marbles from Bags A and B are both green}) \\ &= P(\text{marble from A is green}) * P(\text{marble from B is green}) \\ &= P(A) * P(B) = \frac{1}{2} * \frac{2}{5} = \frac{1}{5} \end{aligned}$$

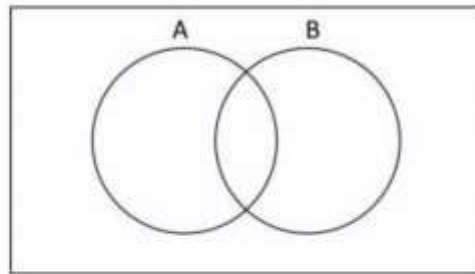
We can multiply $P(A)$ and $P(B)$ because picking from the Bag A and Bag B are independent operations. Picking a marble from Bag A doesn't affect the marbles in Bag B, and picking a marble from Bag B doesn't affect the marbles in Bag A. They are independent.

In general, if events A and B are independent, the equation for $P(A \cap B)$ is

$$P(A \cap B) = P(A) * P(B)$$

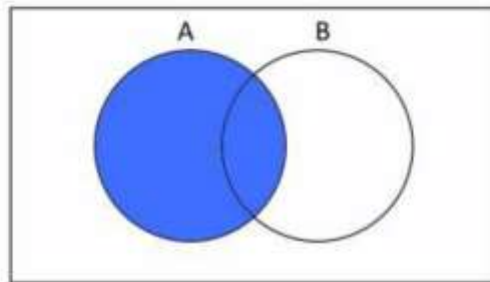
Probability of OR Operations

Let's figure out how to calculate the probability of an OR operation. Let's continue to use events A and B. We will visualize this using a **Venn Diagram**.

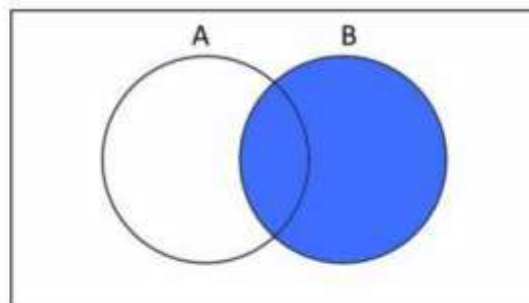


If we want to represent $P(A)$ on this diagram, we would fill in the "A" circle. If we want $P(B)$, we fill in the "B" circle.

$P(A)$

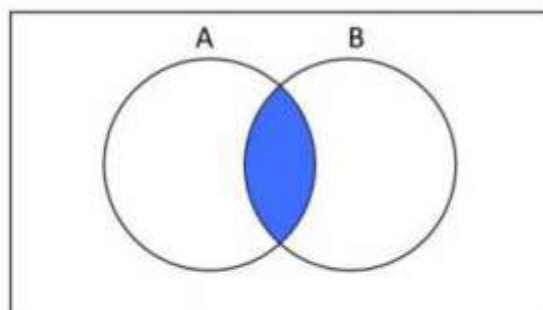


$P(B)$



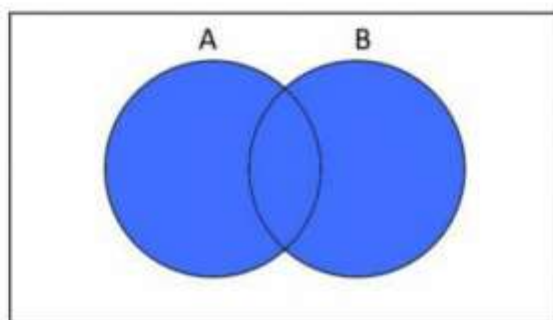
If we want to show the area that represents AND, $P(A \cap B)$, we fill the area that has parts of A and B.

$P(A \cap B)$,



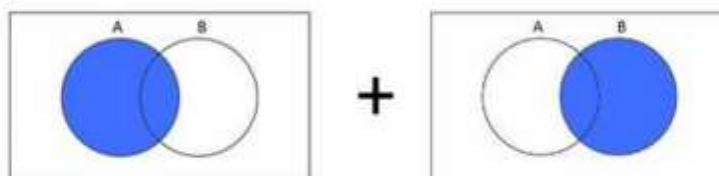
If we want the area for OR, $P(A \cup B)$, we would fill in any space which is inside A or B.

$P(A \cup B)$

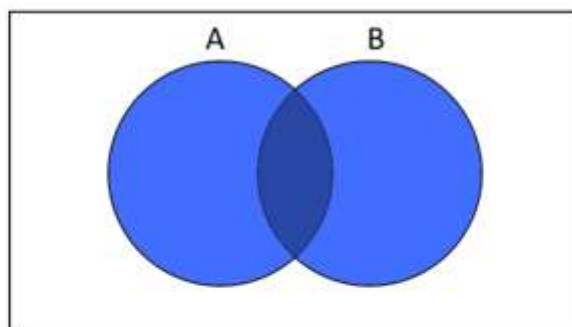


Now let's try to find an equation to calculate $P(A \cup B)$ using Venn Diagrams. We want to fill in the whole space in the circles on the Venn Diagram, so we start by adding $P(A)$ and $P(B)$.

$P(A) + P(B)$

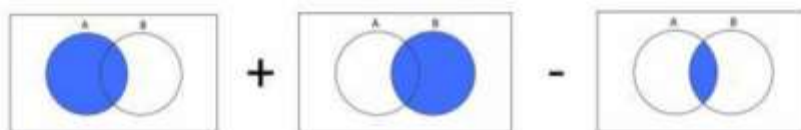


What do you think the result should be? What do you notice is wrong?



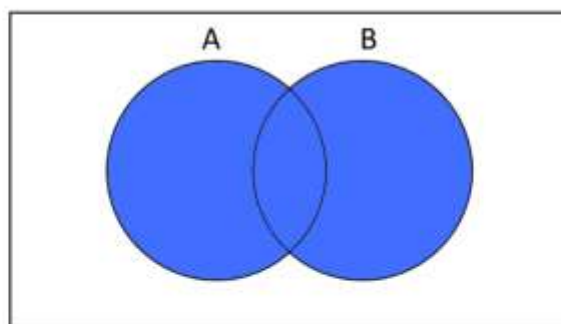
Using this image, we can see that we have counted the middle section twice. We only want to count the middle section once, so we can subtract the middle section to get what we're looking for. The middle section is $P(A \cap B)$, so if we subtract $P(A \cap B)$ we should get $P(A \cup B)$

$$P(A) + P(B) - P(A \cap B)$$



Problems

And now we have our final result, which is $P(A \cup B)$. Using the same



We now know that the general formula for $P(A \cup B)$, for any event A and any event B, is

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Problems



Let's use this equation in practice. We are using the same events A and B from before, where Bag A has 5 green marbles and 5 yellow marbles, and Bag B has 4 green marbles and 6 red marbles. We have calculate that

$$P(A) = \frac{1}{2}$$

$$P(B) = \frac{2}{5}$$

$$P(A \cap B) = \frac{1}{5}$$

So $P(A \cup B) =$ _____

Solutions

Using our equation, we can plug in what we know.

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(A \cup B) = \frac{1}{2} + \frac{2}{5} - \frac{1}{5} = \frac{7}{10}$$

Simulations

Using the Arduino circuit, write down 20 results from picking marbles from the bag.

What percent of the marbles picked from Bag A green? What percent of the marbles picked from Bag B green? What percent of the time are they both green? Does this data match with theoretical probabilities we calculated?

H. What Could Go Wrong

- Make sure the LEDs are placed correctly on the breadboard. You maybe have to reverse the direction to get them to work, due to the positive/negative charge flow.
- Be sure to make sure your buttons are aligned correctly.



I. Challenge – Truth Tables

Truth tables are a good way to understand how AND and OR operations work. They are used in multiple fields such as computer science, mathematics, and any field that uses logic. In the two left columns are the numbers put into the operation, and on the right is the result. Try to fill in the truth tables using what you know about AND and OR. You can look back at the definitions of AND or OR if necessary. Some fields are already filled in.

AND

Argument #1	Argument #2	Result
0	0	
0	1	
1	0	0
1	1	

OR

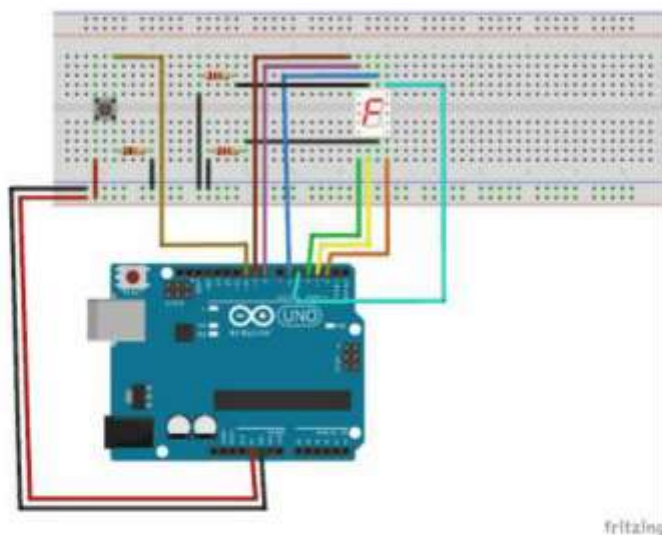
Argument #1	Argument #2	Result
0	0	
0	1	1
1	0	
1	1	

Appendix B



INTRODUCTION TO STATISTICS WITH DICE

[Arduino STEAM]





A. Project Title

Introduction to Statistics with Dice

(Mean, median, and mode)

B. Subjects

Statistics, probability

C. Knowledge

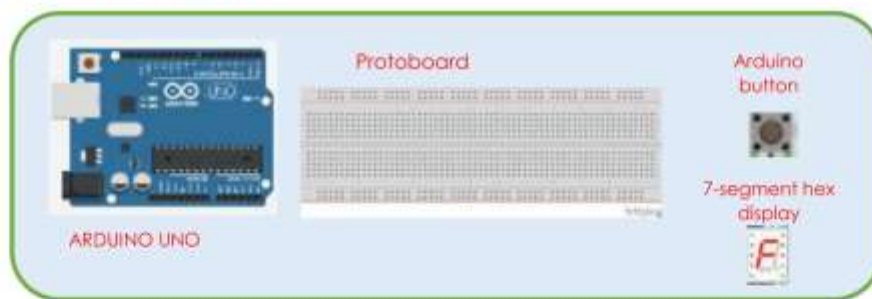
What You Should Know (Prior Knowledge)

- How to wire a circuit
- Resistor color codes
- Arduino IDE

What You Will Learn (Future Knowledge)

- What mean, median, and mode represent
- How to use a 7-segment hex display
- How to use the "random" function in Arduino programming

D. Materials Needed



E. Abstract

Statistics is used in business fields, STEM fields, any job that involves scientific analysis or studies. It is an important concept to understand. Mean, median, and mode are common concepts used in everyday life.

F. Context

A dice factory has a flaw in their most recent design. They want to know if their dice still work well, and can be sold to customers. A working dice should have equal probability of rolling a one, two, three, four, five, and six. Using mean, median, and mode, we will test if the dice are still legitimate.

G. Activity Description

Mean, median, and mode are some of the most useful ways to examine a distribution. They are tools used every day by statisticians, but also by people who



who are not involved in a math field. Mastering these fundamental tools is a great introduction to other math topics. In this lesson, we will also get an introduction to creating and analyzing graphs.

Implementation

We will use the Arduino IDE. Let's examine the "loop" function in our code:

```

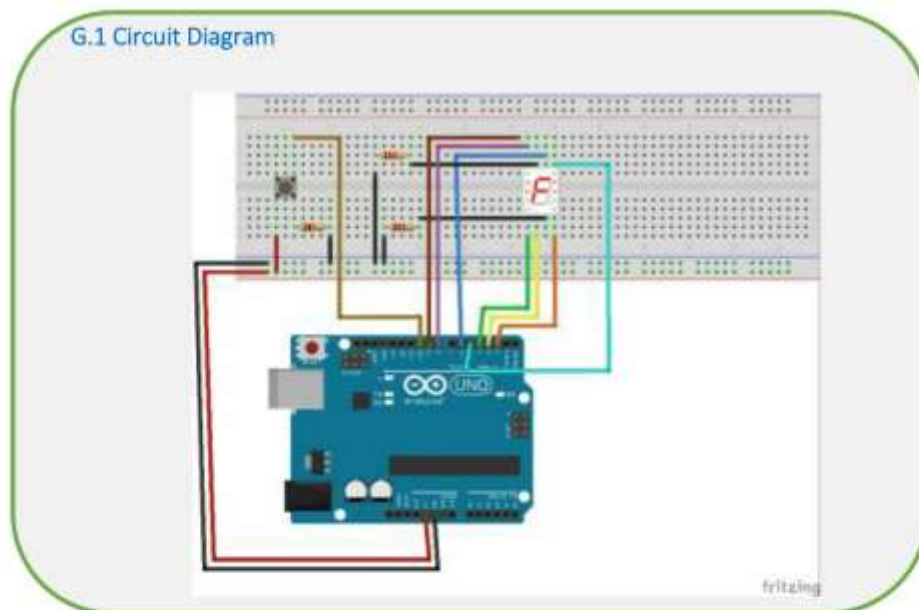
1. void loop() {
2.   if (digitalRead(BUTTON)) {
3.     turnOff();
4.     delay(200);
5.     int randomNumber = random(1, 7);
6.     displayDigit(randomNumber);
7.     delay(250);
8.   }
9. }

```

If the button is pressed, the code inside the if statement will run. The turnoff function will turn off all light on the display. Then it will wait for 200 milliseconds. A random number from 1 to 6 will be generated, and displayed. Then the program will pause for 250 milliseconds, and the loop will run again.

So overall, when a user presses the button, they will see the display go blank, and then see a random number from 1 to 6 displayed.

G.1 Circuit Diagram





G.2 Code

```

1. // Button pin
2. int BUTTON = 10;
3.
4. // Hex pins
5. int a = 6;
6. int b = 5;
7. int c = 2;
8. int d = 3;
9. int e = 4;
10. int f = 8;
11. int g = 9;
12.
13. int hexPins[7] = {a, b, c, d, e, f, g};
14.
15. void setup() {
16.   Serial.begin(9600);
17.   for (int i = 0; i < 7; i++) {
18.     pinMode(hexPins[i], OUTPUT);
19.   }
20. }
21.
22. // changes the hex display to show the "digit" argument
23. void displayDigit(int d) {
24.   if (d != 1 && d != 4){
25.     digitalWrite(a, HIGH);
26.   }
27.   if (d != 5 && d != 6){
28.     digitalWrite(b, HIGH);
29.   }
30.   if (d != 2){
31.     digitalWrite(c, HIGH);
32.   }
33.   if (d != 1 && d != 4 && d != 7){
34.     digitalWrite(d, HIGH);
35.   }
36.   if (d == 2 || d == 6 || d == 8 || d == 0){
37.     digitalWrite(e, HIGH);
38.   }
39.   if (d != 1 && d != 2 && d != 3 && d != 7){
40.     digitalWrite(f, HIGH);
41.   }
42.   if (d != 0 && d != 1 && d != 7){
43.     digitalWrite(g, HIGH);
44.   }
45. }
46.
47. // Turns off the entire display
48. void turnOff() {
49.   for (int i = 0; i < 7; i++) {
50.     digitalWrite(hexPins[i], LOW);
51.   }
52. }
53.
54. void loop() {
55.   if (digitalRead(BUTTON)) {
56.     turnOff();
57.     delay(200);
58.     // Generate a random number from 1 to 6
59.     int randomNumber = random(1, 7);
60.     displayDigit(randomNumber);
61.     delay(250);
62.   }

```



|63.)

G. Activity Description (continued)

Press the button 20 times, and write down the numbers that appear on the hex display. This is your sample distribution.

Mean

The mean (also known as the average) is one method used to measure the center of the distribution. To calculate the mean, add up every number in the distribution and divide by the size of the distribution. Calculate the mean for your dice rolls.

Roll sum: _____

Size of Distribution: 20

Mean = Roll sum / Size of distribution = _____

Median

The median is another method used to measure the center of the distribution. You can calculate the median of the distribution by ordering the distribution from least to greatest and taking the middle value. If the size of the distribution is even, you take the two middle values and average them to get the median.

Using the space below, order your values from least to greatest.

Two middle values: _____, _____

Median: _____

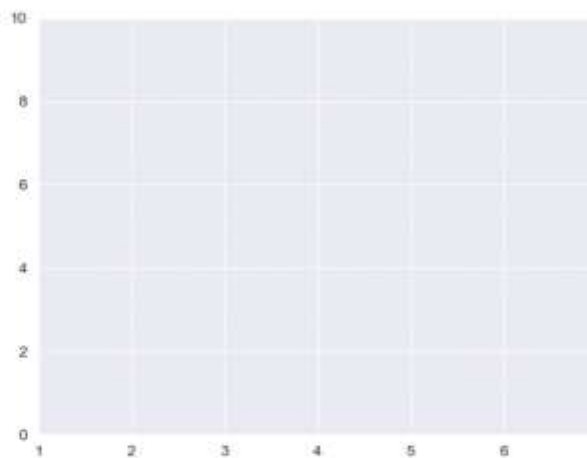
Mode

The mode represents the most likely value to occur in the distribution. In this scenario, the mode is the number most likely to be rolled with the dice. To find the mode, find the number that appears the most times in your sample of 20 numbers.

Mode: _____

Now graph the distribution. On the horizontal axis is the dice number, and on the vertical axis is the number of times that number appeared when you rolled.





Analyze the graph. What do you notice? Does it seem correct, or incorrect? Would you say this dice is working properly? What do you think the graph should look like if there was an equal chance of rolling a 1, 2, 3, 4, 5, or 6?

H. What Could Go Wrong

- It is possible the random numbers generated by the Arduino create a skewed graph. If the graph looks wrong, what do you think it should look like?
- Make sure your 7-segment display and button are placed in the correct positions.

I. Challenge

Take your distribution and add a 100 to it. Recalculate the mean and median.

Mean

Roll sum: _____

Size of Distribution: 21

Mean = Roll sum / Size of distribution = _____

Median

Median: _____

How did the mean change? How did the median change? Why is the median unaffected?





Sine

The sine function takes degrees, and returns the ratio between the opposite leg and the hypotenuse (the longest leg on the triangle).



Let's say you know the angle of the triangle is 30 degrees, and you know the hypotenuse is 6. Now you can use the equation to calculate the length of the opposite leg.

$$\begin{aligned}\sin(a) &= \frac{o}{h} \\ \sin(a) * h &= o \\ \sin(30^\circ) * 6 &= o \\ \frac{1}{2} * 6 &= 3 = o\end{aligned}$$

Sine of 30 degrees is 0.5, so we know the length of the opposite leg of the triangle is 3. You can use your calculator to calculate sine, but make sure it's in degrees mode for this lesson, not radians.

Cosine

The cosine function takes degrees, and returns the ratio between the adjacent leg and the hypotenuse.



We'll use the same example as before. You know the hypotenuse is 6, and the angle is 30°. We will calculate the adjacent angle of the triangle using cosine.

$$\begin{aligned}\cos(a) &= \frac{b}{h} \\ \cos(a) * h &= b \\ \cos(30^\circ) * 6 &= b \\ 0.866 * 6 &= 2.598 = b\end{aligned}$$

Using cosine, we calculated that the length of the adjacent leg to the angle is 2.598.



Calculating the Rocket's Distance

Now you can use the sine and cosine functions to break down the velocity of the rocket into X and Y components. We know that the rocket launches at 50 m/s, and let's suppose the angle is 30 degrees.



Solving for the Y velocity, we get $50 \cdot \sin(30^\circ) = v_y = 25 \text{ m/s}$.



Solving for the X velocity, we get $50 \cdot \cos(30^\circ) = v_x = 43.301 \text{ m/s}$.

Let's look at the free body diagram of the rocket when it's in the air.



There is only one force acting on the rocket: gravity. This means that when the rocket is in the air, the X velocity will be constant, and the Y velocity will always be changing. Since the X velocity is constant, we can calculate the distance the rocket travels by multiplying the X velocity times the amount of time the rocket is in the air.

$$\text{Distance} = v_x t_a$$

We know the X velocity is 43.301, so we just need to find t_a . To do this, we can use a common equation in mechanics.

$$x = x_0 + v_0 t + \frac{1}{2} a t^2$$

In this equation, x represents the x coordinate at a particular time t . x_0 represents the initial position on the axis. v_0 is the initial velocity on the axis. a is the acceleration on the axis.

In this case we can change the "x"s to "y"s since we are working with the y axis.

$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$





Each of these variables should only be related to the y-axis. For example instead of using 50 as the initial y velocity, we would only use the Y component of the velocity.

Remember, we want to find the time in the air. What happens to the y position when the rocket is no longer in the air?

The answer is that the y position is 0, since at 0 it hits the ground, and is no longer in the air. So now we know that to get t_a , we need to solve for when y is 0.

$$0 = y_0 + v_0 t_a + \frac{1}{2} a t_a^2$$

Now we have to figure out y_0 , v_0 , and a . What is the original y position of the rocket? It starts on the ground, which means $y = 0$. Therefore $y_0 = 0$.

What is the original velocity on the y-axis? Remember, we cannot use 50, since that includes the x component. We must use the y-component of the velocity, which we calculated as 25 m/s.

What is the acceleration of the rocket on the y-axis? There is only one force acting on the rocket, which is gravity. Gravity works in the downward direction (negative), and the rocket is flying in the upwards direction (positive). Therefore the acceleration is -9.8 m/s^2 .

Now we can fill in the variables.

$$0 = 0 + 25 * t_a + \frac{1}{2} * -9.8 * t_a^2$$

We can now solve for t_a .

$$0 = 25 * t_a - 4.9 * t_a^2$$

$$4.9 * t_a^2 = 25 * t_a$$



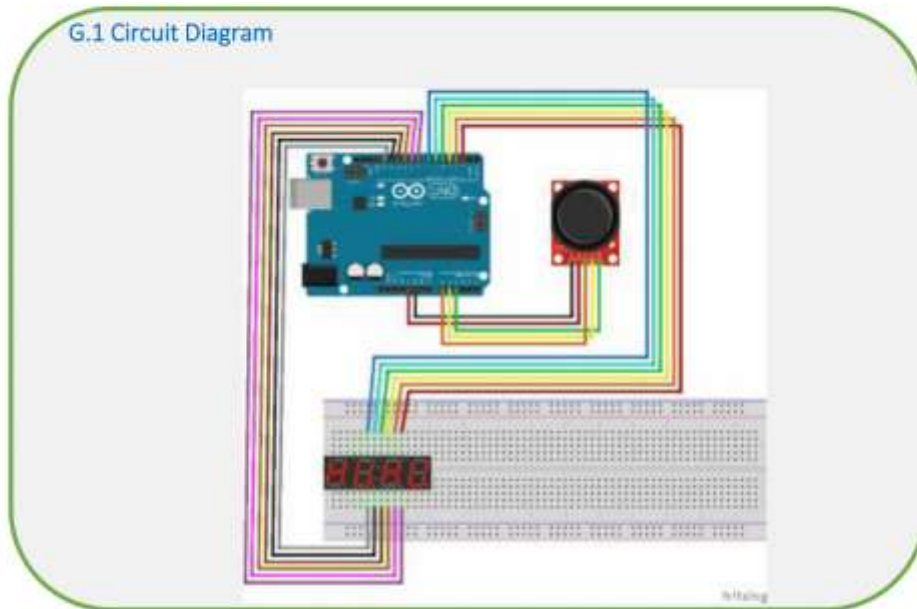
$$t_a = \frac{25}{4.9} = 5.102$$

We have now calculated t_a , so we can go back to our equation to solve for distance.

$$\text{Distance} = v_x t_a = 43.301 \times 5.102$$

$$\text{Distance} = 220.925 \text{ meters}$$

G.1 Circuit Diagram





G.2 Code

```

1. #include "SevSeg.h"
2.
3. SevSeg sevseg;
4.
5. const int X_pin = A0;
6. const int Y_pin = A1;
7. const int SW_pin = A2;
8.
9.
10. float angle = 20.0f;
11.
12. byte numDigits = 4;
13. byte digitPins[] = {7, 4, 3, 8};
14. byte segmentPins[] = {6, 2, 10, 12, 13, 5, 9, 11};
15.
16. void setup() {
17.   Serial.begin(9600);
18.
19.   sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins);
20.
21.   sevseg.setBrightness(90);
22.
23.   pinMode(SW_pin, INPUT);
24.   digitalWrite(SW_pin, HIGH);
25.
26.   sevseg.setNumber(round(angle*100), 2);
27.
28. }
29.
30. float degrees_to_radians(float angle_d) {
31.   return angle_d / 360.0f * 2 * PI;
32. }
33.
34.
35. void delay_while_refresh(long wait_time) {
36.   long start = millis();
37.   while (millis() - start < wait_time) {
38.     sevseg.refreshDisplay();
39.   }
40. }
41.
42. float m = 50;
43. float g = 9.8f;
44. float simulate_launch(float angle_d) {
45.   float angle_r = degrees_to_radians(angle_d);
46.   float orig_x_vel = m * cos(angle_r);
47.   float orig_y_vel = m * sin(angle_r);
48.   float time_of_max_height = orig_y_vel / g;
49.   float max_height = orig_y_vel * time_of_max_height - 0.5 * g *
   time_of_max_height * time_of_max_height;
50.   float time_in_air = orig_y_vel * 2 / g;
51.   float distance_traveled = time_in_air * orig_x_vel;
52.
53.   float eq_a = -4 * max_height / (distance_traveled * distance_traveled);
54.   float eq_b = max_height * 4 / distance_traveled;
55.
56.   for (int i = 0; i < distance_traveled; i++) {
57.     Serial.println(eq_a * i * i + eq_b * i);
58.     delay_while_refresh(25);
59.   }
60.   return distance_traveled;
61. }

```



```

62.
63. void check_angle_boundaries(){
64.   angle = max(0, angle);
65.   angle = min(90, angle);
66. }
67.
68. void loop() {
69.
70.   if (digitalRead(SW_pin)) {
71.     float d = simulate_launch(angle);
72.     sevseg.setNumber(d, -1);
73.   }
74.
75.   int joystick_x = analogRead(X_pin);
76.   if (joystick_x < 450) {
77.     angle -= 0.5;
78.     check_angle_boundaries();
79.     sevseg.setNumber(round(angle * 100), 2);
80.   } else if (joystick_x > 550) {
81.     angle += 0.5;
82.     check_angle_boundaries();
83.     sevseg.setNumber(round(angle * 100), 2);
84.   }
85.
86.   delay_while_refresh(100);
87. }

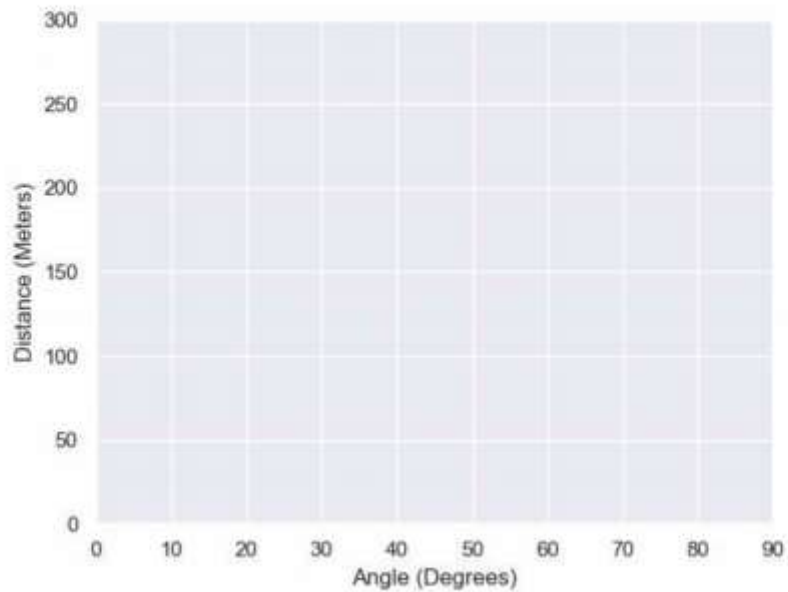
```

G. Activity Description (continued)

Open the Arduino IDE, and enter the code. Upload it to the Arduino, and then move the joystick until the display shows "30". This number represents the angle you shoot the rocket at. Now open the Arduino plotter by clicking "Tools -> Serial Plotter".

Press down the joystick to launch the rocket, and you will see the rocket's path displayed on the serial plotter. Once the rocket has landed, the distance it traveled will be displayed on the hex display. Double check to make sure the distance traveled is the same as what we calculated (220.925 meters).

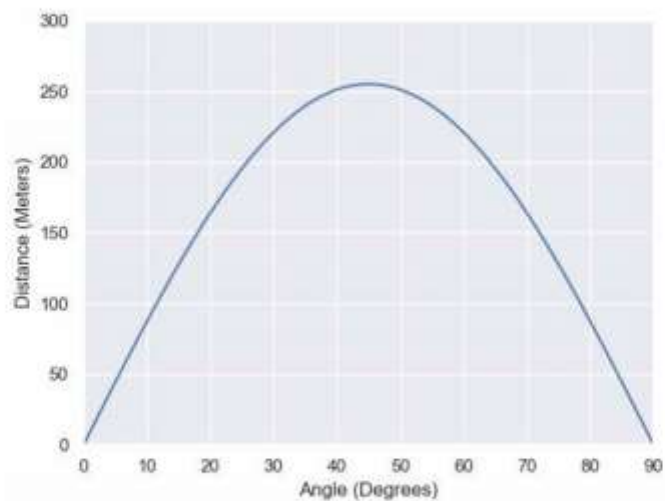
How do you make sure the rocket gets to the other side of the field? You need to find the angle that makes the rocket travel the farthest.



Try 15 angles, and plot them on the graph above. The X axis is the angle you used, and the Y axis is the distance traveled. Try to choose angles that come from all parts of the X axis.

Now analyze your data. What angle do you think launches the rocket the farthest. Try to draw a line or curve that represents the data.

Solution





The maximum distance can be achieved with a 45 degree angle. Note that this is exactly halfway between 0 and 90. Why do you think 35 degrees gives the maximum distance? Provide your reasoning.

H. What Could Go Wrong

- Make sure your 4 7-segment display is oriented properly.
- If the joystick does not respond, see if you have the same type of joystick. Match the GRD, 5 V, and other labels with the correct wires in the diagram.

I. Challenge

Let's say the rocket is put on a cliff before being launched. What do you think the angle would be to get the maximum distance traveled? Explain your intuition for your answer.

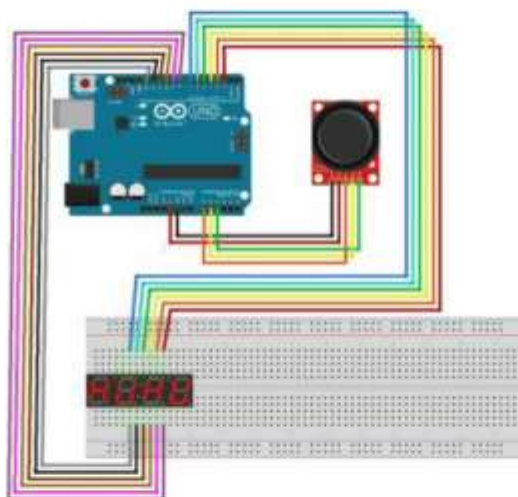


Appendix C



PHYSICS WITH ROCKET LAUNCHES

[Arduino STEAM]



nylalyq



A. Title of Project

Physics with Rocket Launches

(Angles, basic physics formulas)

B. Subjects

Physics, trigonometry.

C. Knowledge

What You Should Know (Prior Knowledge)

- How to wire a circuit
- Arduino IDE
- Position, velocity, and acceleration.
- Pythagorean theorem
- Algebra
- Free body diagrams

What You Will Learn (Future Knowledge)

- Angles, sine, cosine
- Basic mechanical formulas and concepts

D. Materials Needed



E. Abstract

The lessons learned in physics often are useful in other fields. Conducting studies, figuring out how to manipulate equations, and understanding a situation using equations are all useful skills to have in a STEM field.

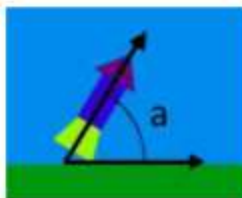
F. Context

You have a rocket that you are launching in a field. You want it to get to the other side of the field, but you don't know what angle to place the rocket at. How do you make sure the rocket goes as far as it can? This problem involves physics.

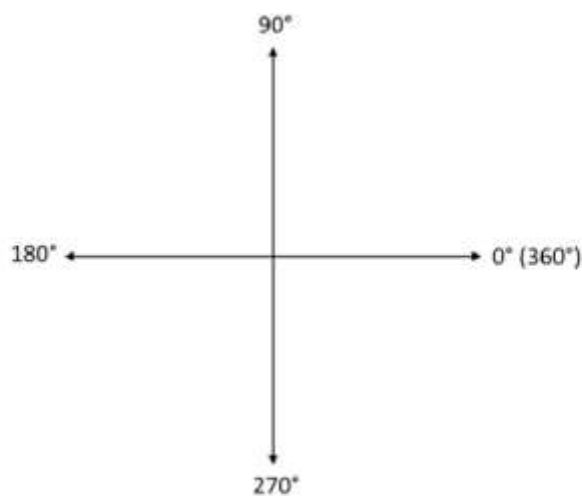


G. Activity Description

An angle is a way to measure the space between two lines that meet at a point. For example, in the image shown, the angle is represented using "a".



Angles are measured using degrees. Using the degrees measurement, a full circle is 360 degrees. A half circle is 180 degrees. The following is a diagram to show how degrees work. The symbol for degrees is $^{\circ}$.





To get the maximum distance, you will have to fire your rocket somewhere between 0 and 90 degrees.

Mechanical Physics

The rocket is launched at 50 m/s (meters per second) no matter the angle chosen. However, this velocity can be split into an X velocity and a Y velocity (the X and Y axes). For example, if you launch the rock at 30 degrees, the rocket is going up so there must be a Y velocity, but it's also going forward, so there must be an X velocity.

For example, let's say you're given an angle. You can draw a horizontal and a vertical line to turn this angle into a triangle.

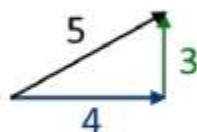


The green arrow represents the Y velocity, and the blue arrow represents the X velocity.



Using what you know about the Pythagorean theorem, you can calculate the length of the black line.

$$\begin{aligned}x^2 &= 3^2 + 4^2 \\x^2 &= 25 \\x &= 5\end{aligned}$$



Now we want to find the X and Y velocities of the rocket, since we know the angle and we know that it is launched at 50 m/s. To do this we need to use sine and cosine.

Sine and cosine are mathematical functions which help you calculate the ratios between legs on a right triangle.



Sine

The sine function takes degrees, and returns the ratio between the opposite leg and the hypotenuse (the longest leg on the triangle).



Let's say you know the angle of the triangle is 30 degrees, and you know the hypotenuse is 6. Now you can use the equation to calculate the length of the opposite leg.

$$\begin{aligned}\sin(a) &= \frac{o}{h} \\ \sin(a) * h &= o \\ \sin(30^\circ) * 6 &= o \\ \frac{1}{2} * 6 &= 3 = o\end{aligned}$$

Sine of 30 degrees is 0.5, so we know the length of the opposite leg of the triangle is 3. You can use your calculator to calculate sine, but make sure it's in degrees mode for this lesson, not radians.

Cosine

The cosine function takes degrees, and returns the ratio between the adjacent leg and the hypotenuse.



We'll use the same example as before. You know the hypotenuse is 6, and the angle is 30°. We will calculate the adjacent angle of the triangle using cosine.

$$\begin{aligned}\cos(a) &= \frac{b}{h} \\ \cos(a) * h &= b \\ \cos(30^\circ) * 6 &= b \\ 0.866 * 6 &= 2.598 = b\end{aligned}$$

Using cosine, we calculated that the length of the adjacent leg to the angle is 2.598.



Calculating the Rocket's Distance

Now you can use the sine and cosine functions to break down the velocity of the rocket into X and Y components. We know that the rocket launches at 50 m/s, and let's suppose the angle is 30 degrees.



Solving for the Y velocity, we get $50 \cdot \sin(30^\circ) = v_y = 25 \text{ m/s}$.



Solving for the X velocity, we get $50 \cdot \cos(30^\circ) = v_x = 43.301 \text{ m/s}$.

Let's look at the free body diagram of the rocket when it's in the air.



There is only one force acting on the rocket: gravity. This means that when the rocket is in the air, the X velocity will be constant, and the Y velocity will always be changing. Since the X velocity is constant, we can calculate the distance the rocket travels by multiplying the X velocity times the amount of time the rocket is in the air.

$$\text{Distance} = v_x t_a$$

We know the X velocity is 43.301, so we just need to find t_a . To do this, we can use a common equation in mechanics.

$$x = x_0 + v_0 t + \frac{1}{2} a t^2$$

In this equation, x represents the x coordinate at a particular time t . x_0 represents the initial position on the axis. v_0 is the initial velocity on the axis. a is the acceleration on the axis.

In this case we can change the "x"s to "y"s since we are working with the y axis.

$$y = y_0 + v_0 t + \frac{1}{2} a t^2$$





Each of these variables should only be related to the y-axis. For example instead of using 50 as the initial y velocity, we would only use the Y component of the velocity.

Remember, we want to find the time in the air. What happens to the y position when the rocket is no longer in the air?

The answer is that the y position is 0, since at 0 it hits the ground, and is no longer in the air. So now we know that to get t_a , we need to solve for when y is 0.

$$0 = y_0 + v_0 t_a + \frac{1}{2} a t_a^2$$

Now we have to figure out y_0 , v_0 , and a . What is the original y position of the rocket? It starts on the ground, which means $y = 0$. Therefore $y_0 = 0$.

What is the original velocity on the y-axis? Remember, we cannot use 50, since that includes the x component. We must use the y-component of the velocity, which we calculated as 25 m/s.

What is the acceleration of the rocket on the y-axis? There is only one force acting on the rocket, which is gravity. Gravity works in the downward direction (negative), and the rocket is flying in the upwards direction (positive). Therefore the acceleration is -9.8 m/s^2 .

Now we can fill in the variables.

$$0 = 0 + 25 * t_a + \frac{1}{2} * -9.8 * t_a^2$$

We can now solve for t_a .

$$0 = 25 * t_a - 4.9 * t_a^2$$

$$4.9 * t_a^2 = 25 * t_a$$



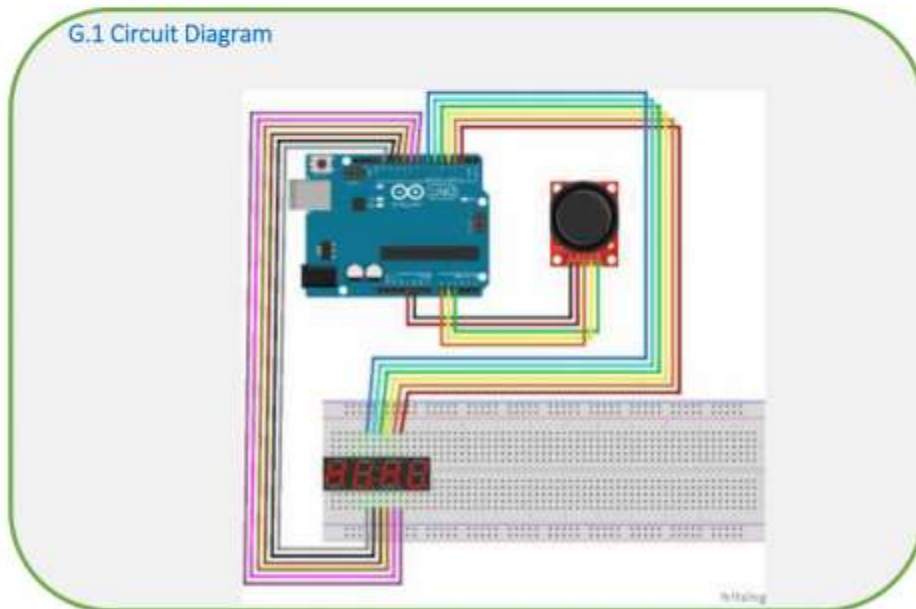
$$t_a = \frac{25}{4.9} = 5.102$$

We have now calculated t_a , so we can go back to our equation to solve for distance.

$$\text{Distance} = v_x t_a = 43.301 \cdot 5.102$$

$$\text{Distance} = 220.925 \text{ meters}$$

G.1 Circuit Diagram





G.2 Code

```

1. #include "SevSeg.h"
2.
3. SevSeg sevseg;
4.
5. const int X_pin = A0;
6. const int Y_pin = A1;
7. const int SW_pin = A2;
8.
9.
10. float angle = 20.0f;
11.
12. byte numDigits = 4;
13. byte digitPins[] = {7, 4, 3, 8};
14. byte segmentPins[] = {6, 2, 10, 12, 13, 5, 9, 11};
15.
16. void setup() {
17.   Serial.begin(9600);
18.
19.   sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins);
20.
21.   sevseg.setBrightness(90);
22.
23.   pinMode(SW_pin, INPUT);
24.   digitalWrite(SW_pin, HIGH);
25.
26.   sevseg.setNumber(round(angle*100), 2);
27.
28. }
29.
30. float degrees_to_radians(float angle_d) {
31.   return angle_d / 360.0f * 2 * PI;
32. }
33.
34.
35. void delay_while_refresh(long wait_time) {
36.   long start = millis();
37.   while (millis() - start < wait_time) {
38.     sevseg.refreshDisplay();
39.   }
40. }
41.
42. float m = 50;
43. float g = 9.8f;
44. float simulate_launch(float angle_d) {
45.   float angle_r = degrees_to_radians(angle_d);
46.   float orig_x_vel = m * cos(angle_r);
47.   float orig_y_vel = m * sin(angle_r);
48.   float time_of_max_height = orig_y_vel / g;
49.   float max_height = orig_y_vel * time_of_max_height - 0.5 * g *
time_of_max_height * time_of_max_height;
50.   float time_in_air = orig_y_vel * 2 / g;
51.   float distance_traveled = time_in_air * orig_x_vel;
52.
53.   float eq_a = -4 * max_height / (distance_traveled * distance_traveled);
54.   float eq_b = max_height * 4 / distance_traveled;
55.
56.   for (int i = 0; i < distance_traveled; i++) {
57.     Serial.println(eq_a * i * i + eq_b * i);
58.     delay_while_refresh(25);
59.   }
60.   return distance_traveled;
61. }

```




```

62.
63. void check_angle_boundaries(){
64.   angle = max(0, angle);
65.   angle = min(90, angle);
66. }
67.
68. void loop() {
69.
70.   if (!digitalRead(SW_pin)) {
71.     float d = simulate_launch(angle);
72.     sevseg.setNumber(d, -1);
73.   }
74.
75.   int joystick_x = analogRead(X_pin);
76.   if (joystick_x < 450) {
77.     angle -= 0.5;
78.     check_angle_boundaries();
79.     sevseg.setNumber(round(angle * 100), 2);
80.   } else if (joystick_x > 550) {
81.     angle += 0.5;
82.     check_angle_boundaries();
83.     sevseg.setNumber(round(angle * 100), 2);
84.   }
85.
86.   delay_while_refresh(100);
87. }

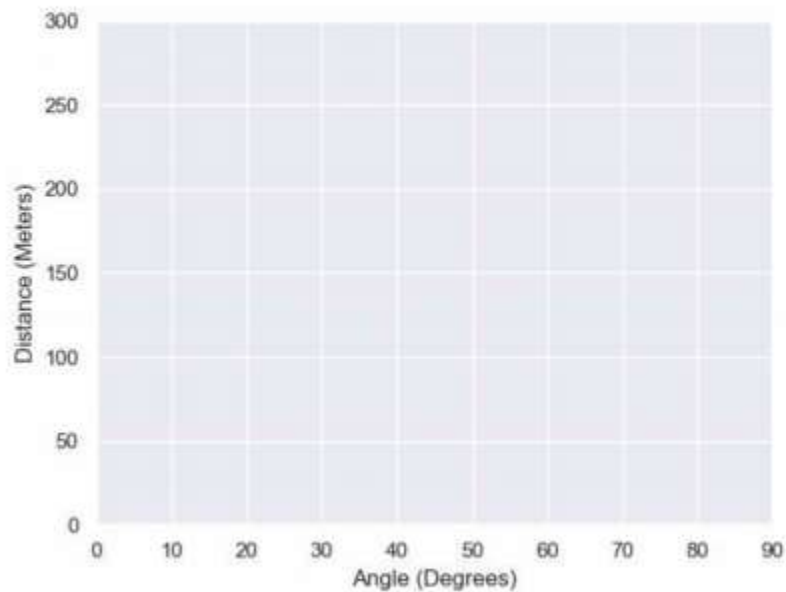
```

G. Activity Description (continued)

Open the Arduino IDE, and enter the code. Upload it to the Arduino, and then move the joystick until the display shows "30". This number represents the angle you shoot the rocket at. Now open the Arduino plotter by clicking "Tools -> Serial Plotter".

Press down the joystick to launch the rocket, and you will see the rocket's path displayed on the serial plotter. Once the rocket has landed, the distance it traveled will be displayed on the hex display. Double check to make sure the distance traveled is the same as what we calculated (220.925 meters).

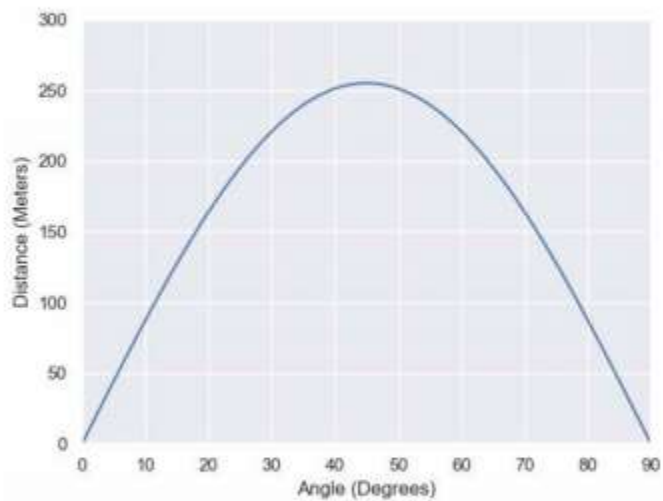
How do you make sure the rocket gets to the other side of the field? You need to find the angle that makes the rocket travel the farthest.



Try 15 angles, and plot them on the graph above. The X axis is the angle you used, and the Y axis is the distance traveled. Try to choose angles that come from all parts of the X axis.

Now analyze your data. What angle do you think launches the rocket the farthest. Try to draw a line or curve that represents the data.

Solution





The maximum distance can be achieved with a 45 degree angle. Note that this is exactly halfway between 0 and 90. Why do you think 45 degrees gives the maximum distance? Provide your reasoning.

H. What Could Go Wrong

- Make sure your 4 7-segment display is oriented properly.
- If the joystick does not respond, see if you have the same type of joystick. Match the GRD, 5 V, and other labels with the correct wires in the diagram.

I. Challenge

Let's say the rocket is put on a cliff before being launched. What do you think the angle would be to get the maximum distance traveled? Explain your intuition for your answer.



Appendix D



REVERSE ENGINEERING A KEYPAD CALCULATOR

[Arduino STEAM]





A. Project Title

Keypad Calculator

B. Subjects

Reverse engineering, computer programming, self-teaching

C. Knowledge

What You Should Know (Prior Knowledge)

- Arduino IDE
- if statements, arrays, boolean operators

What You Will Learn (Future Knowledge)

- Functions
- How to program a state machine
- How to use the Serial Monitor
- How to reverse engineer a machine

D. Materials Needed



E. Abstract

A large amount of work conducted in any field in the real world involves examining and understanding other peoples' work. Reverse engineering is a difficult skill to develop, but it is vital if you are put in a situation where you have to learn from past work done by other people.

F. Context

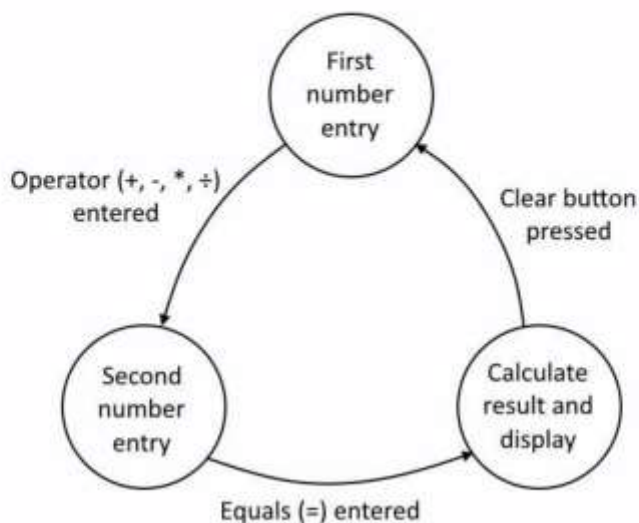
Your school needs to be able to make more calculators. They have one, but no one can understand the code. It's your job to decipher it and find out what the code does!

G. Activity Description

A calculator seems like a very basic, everyday object, but programming a calculator can be quite difficult! A calculator is a finite state machine, which is a



model of a machine that has several states, where a user can give input, and the machine changes from one state to the next.



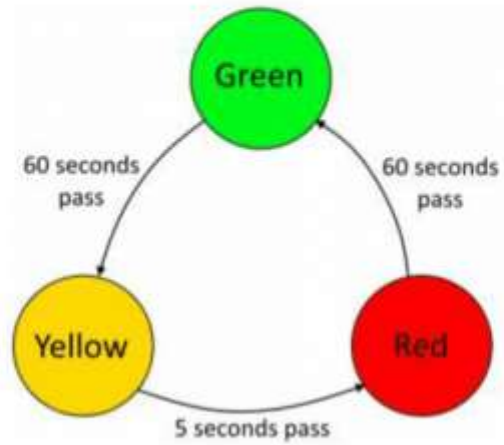
Each circle represents a machine state, and each arrow represents user input to change the calculator from one state to the next. For example, when you enter numbers on the calculator, it is taking your input and appending each digit you enter to the last. When you enter an operator such as plus, minus, multiply and divide, it changes the state to taking in the second number in your equation. It continues to be in this state until you enter an equals sign.

After the equals sign is entered, the calculator changes states to be in the next state, where it calculates the result of your equation and displays it. It will remain in this state until you press "Clear", and then it is back in the first state, where it is now taking digits for you to enter.

This is what a finite state machine is. It has several states it can be in, and user input changes the states so the machine can perform useful operations.



A simpler example of a state machine is a traffic light.



The traffic light starts in the Green state, and stays there for 60 seconds. After Green it moves to Yellow, and then to Red, and then back to Green.

G.1 Circuit Diagram





G.2 Code

```

1. #include <ArduinoSTL.h>
2.
3. #include <Key.h>
4. #include <Keypad.h>
5.
6. const byte ROWS = 4;
7. const byte COLS = 4;
8.
9. char hexaKeys[ROWS][COLS] = {
10.  {'1', '2', '3', 'A'},
11.  {'4', '5', '6', 'B'},
12.  {'7', '8', '9', 'C'},
13.  {'*', '0', '#', 'D'}
14. };
15.
16. byte rowPins[ROWS] = {9, 8, 7, 6};
17. byte colPins[COLS] = {5, 4, 3, 2};
18.
19. Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
    COLS);
20.
21. void setup() {
22.   Serial.begin(9600);
23. }
24.
25. long firstNumber = 0;
26. long secondNumber = 0;
27. char op = ' ';
28.
29. void reset() {
30.   firstNumber = 0;
31.   secondNumber = 0;
32.   op = ' ';
33. }
34.
35. bool is_number(char key) {
36.   return '0' <= key && key <= '9';
37. }
38.
39. bool is_op(char key) {
40.   return key == 'A' || key == 'B' || key == 'D' || key == '*';
41. }
42.
43. bool is_clear(char key) {
44.   return key == 'C';
45. }
46.
47. bool is_equals(char key) {
48.   return key == '#';
49. }
50.
51. long add_digit_to_end_of_num(long num, char digitChar) {
52.   long digit = digitChar - '0';
53.   return num * 10L + digit;
54. }
55.
56.
57. char opToSymbol(char op) {
58.   if (op == 'A') {
59.     return '+';
60.   }
61.   else if (op == '*') {

```




```

62.     return '*';
63. }
64. else if (op == '0') {
65.     return '/';
66. }
67. else if (op == 'B') {
68.     return '-';
69. }
70. return '?';
71. }
72.
73. void performOp(long firstNum, char op, long secondNum) {
74.     if (op == '+') {
75.         Serial.println(firstNum + secondNum);
76.     }
77.     else if (op == '*') {
78.         Serial.println(firstNum * secondNum);
79.     }
80.     else if (op == '/') {
81.         Serial.println(static_cast<double>(firstNum) * 1.0 / secondNum);
82.     }
83.     else if (op == '-') {
84.         Serial.println(firstNum - secondNum);
85.     }
86. }
87.
88. void loop() {
89.     char key = customKeypad.getKey();
90.     if (key) {
91.         if (is_number(key)) {
92.             if (op == ' ') {
93.                 firstNumber = add_digit_to_end_of_num(firstNumber, key);
94.             } else {
95.                 secondNumber = add_digit_to_end_of_num(secondNumber, key);
96.             }
97.             Serial.print(key);
98.         }
99.         else if (is_op(key)) {
100.             op = opToSymbol(key);
101.             Serial.print(op);
102.         }
103.         else if (is_clear(key)) {
104.             reset();
105.             Serial.print("\n---\n");
106.         }
107.         else if (is_equals(key)) {
108.             Serial.print("=");
109.             performOp(firstNumber, op, secondNumber);
110.             reset();
111.         }
112.     }
113. }

```

G. Activity Description (continued)

Open the Arduino IDE, and enter the given code. Upload the code to the Arduino, and then open the Arduino Serial Monitor. When you press a button the keypad, it should appear on the serial monitor. Note the following keys perform the operations:

- Any number key is the number
- "A" is used for addition
- "B" is used for subtraction
- "C" is used to clear the calculator



- "D" is used for division
- "*" is for multiplication
- "=" is the equals key

For example, if you want to know 5 plus 2, type 5, then type A, then type 2, and then #. Play with the calculator, and make sure you understand how it works.

First examine the smaller functions, such as `is_number`, `is_op`, `is_clear`, `is_equals`. Using your knowledge of how the calculator works, figure out what these functions do. For example, the `is_number` function checks to see if a character given from the keypad represents a number. The `is_op` function checks to see if the given character is an operator character such as A, B, D, and *.

On your own, try to understand the other functions in the code, and dissect the loop function. Write down what each function does, and try to get an understanding of everything. Make sure you keep in mind the knowledge you have about what the calculator does. This will help you understand the code.

Once you are done writing down the purpose of each function, try to write down what the loop function does. Keep in mind that the calculator is a state machine.

If you have trouble, fill in the following problems.

1. If the calculator is inputting the first number, then _____ will be _____.
2. If the calculator is inputting the second number, then _____ will be _____.

Solutions

To program a state machine, you need to be able to check which state the machine is in. For example, if a user enters a digit, how do we know if the digit is for the first number or the second? The answer is using the state of the operator.

1. If the calculator is inputting the first number, then **the operator** will be **blank**.
2. If the calculator is inputting the second number, then **the operator** will be **not blank**.

Note that the operator is represented with the "op" variable. If the first number is being inputted, the operator will be blank because the first number comes before the operator is entered. If the second number is being entered, the operator will not be blank because the second number is entered after the operator.

Note that this is a way to determine which state the calculator is in (Look back at the state machine diagram if necessary).

H. What Could Go Wrong

- If you get stuck, take a quick look at the solution, but don't read the whole thing. You may want to review basic programming concepts before you complete the activity.
- If the keypad is unresponsive, try using different wires. If that does not work, the keypad may be broken.



I. Challenge

See if you can add your own idea to the code. Come up with a new operation to replace addition (the "A" key), and implement it. Change what the calculator does when the addition operation is performed, and see if it works!

Challenge Example

Let's say you want the A key to put the two numbers together without adding them. For example you enter "1 A 5" and you get 15, joining 1 and 5 together. Or you enter "1234 A 5678", then you get "12345678". You can do this easily using `Serial.print`. Using `Serial.print`, you would print the first number without a new line, and then print the second number, with a new line afterwards to indicate you are done.

The only code you would need to change to implement this function is the "performOp" function. Look at the changes below, in the lines below the "op == '+'" line:

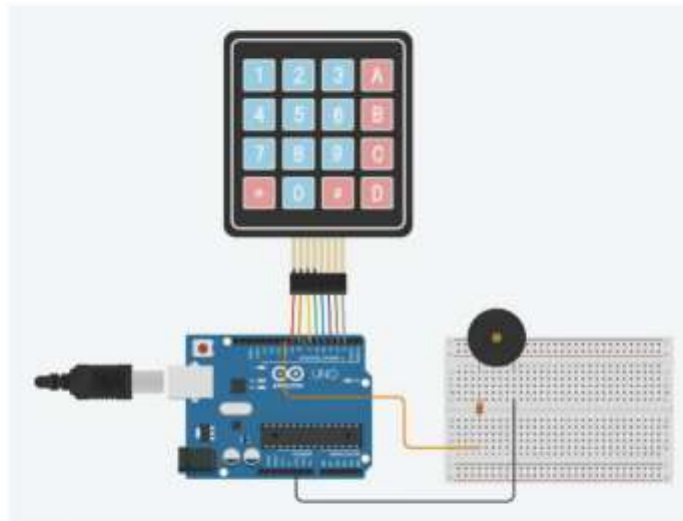
```
1. void performOp(long firstNum, char op, long secondNum) {
2.   if (op == '+') {
3.     Serial.print(firstNum);
4.     Serial.println(secondNum);
5.   }
6.   else if (op == '*') {
7.     Serial.println(firstNum * secondNum);
8.   }
9.   else if (op == '/') {
10.    Serial.println(static_cast<double>(firstNum) * 1.0 / secondNum);
11.  }
12.  else if (op == '-') {
13.    Serial.println(firstNum - secondNum);
14.  }
15. }
```

Appendix E



THE PHYSICS BEHIND MUSIC

[Arduino STEAM]





A. Title of Project

The Physics Behind Music
(Frequency, wavelength, wave formulas)

B. Subjects

Physics, waves, math

C. Knowledge

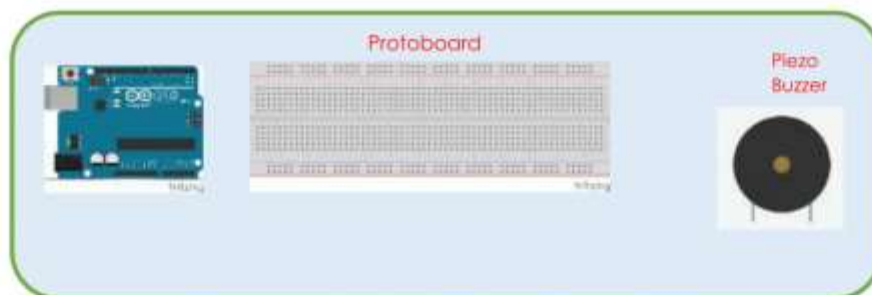
What You Should Know (Prior Knowledge)

- How to wire a circuit
- Resistors
- Arrays, functions

What You Will Learn (Future Knowledge)

- How music is made
- Frequency, wavelength, wave formulas
- How to use the Piezo Buzzer

D. MATERIALS NEEDED



E. Abstract

Waves appear everywhere in our life, but are often unseen. Light waves, radio waves, sound waves are all waves that we can perceive, but we do not immediately recognize them as waves. This workshop teaches the basic properties of waves.

F. Context

You have been assigned the task of creating a keypad for a safe, where the code to unlock the safe can be memorized through a song. Each key on the keypad produces a unique pitch, and a series of pitches can be translated into the safes password.

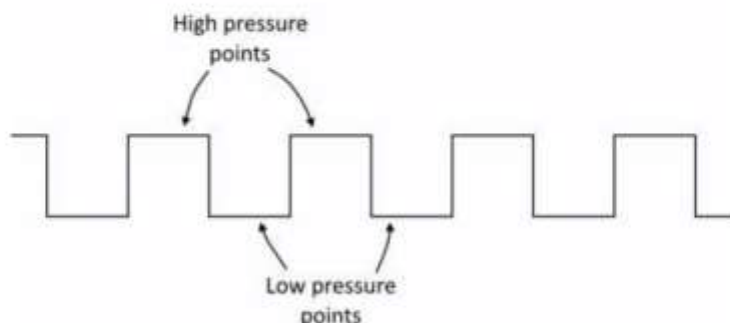
G. Activity Description

Sound is produced by vibrations in air. Vibrations cause areas in air where there are high and low pressures, which in turn produces a wave. There are many



different types of waves. For example, the Piezo Buzzer produces a square wave, which is a wave with only high and low pressure areas, with nothing in between.

Square Wave



Another type of wave is a sine wave, which sounds similar to an organ. Every wave has a property called a period. The period of the wave is the time it takes to complete its cycle, in seconds. For example, in a square wave, the period is the time it takes to get from one upward point to another upward point. Period is represented using a capital T.

Square Wave



The frequency of a wave is the number of cycles the wave completes per second. The unit for frequency is $\text{Hz} = \frac{1}{s}$, where the denominator is seconds. Frequency is often used in music. For example, a common tuning for a musical instrument is based on the fact that A4, a musical note, is equal to 440 Hz. This means that a perfectly in tune A4 note completes 440 wave cycles per second.

Period and frequency are closely related. If the period of a wave increases, the frequency decreases. If the period decreases, the frequency increases. The formula to convert from period to frequency is

$$f = \frac{1}{T}$$



Where T is the waves period and f is the waves frequency.

Wavelength is another commonly used property to describe a wave.

Wavelength is the distance a wave travels in one period, measured in meters. If we let v represent the velocity of the wave, we can calculate wavelength with the following formula.

$$\lambda = v \cdot T$$

Where λ represents the wavelength. By multiplying v , the velocity, by T , the period in seconds, you get a distance. This is the distance that the wave traveled in one period.

The speed of sound is know to be $v = 242 \frac{m}{s}$. Solve the following problems.

Problems

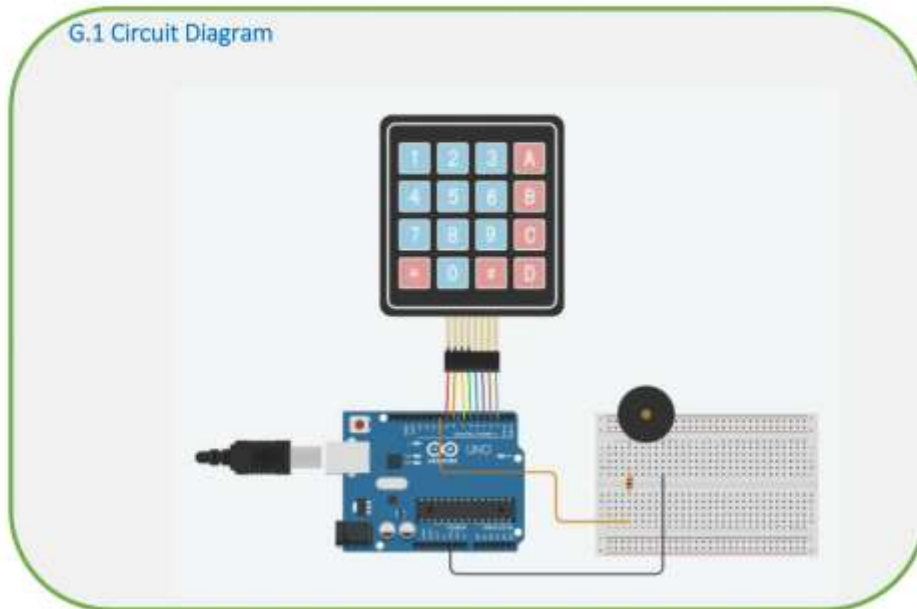
1. If $T = 0.1$ seconds, $f =$ _____ Hz.
2. If $f = 100$ Hz, $T =$ _____ seconds.
3. If $T = 0.01$ seconds, $v =$ _____ $\frac{m}{s}$.
4. If $f = 500$ Hz, $\lambda =$ _____ m.
5. If $T = 10$ seconds, $\lambda =$ _____ m.
6. If $\lambda = 484$ m, $f =$ _____ Hz.

Solutions

1. $f = 10$ Hz
2. $T = \frac{1}{100}$ seconds = 0.01 seconds
3. $v = 242 \frac{m}{s}$
4. $\lambda = v \cdot T = \frac{v}{f} = \frac{242}{500} = 0.484$ meters
5. $\lambda = v \cdot T = 242 \cdot 10 = 2420$ meters
6. $484 = 242 \cdot T = \frac{242}{f}$. $f = \frac{242}{484} = 0.5$ Hz.



G.1 Circuit Diagram



G.2 Code

```

1. #include <ArduinoSTL.h>
2. #include <Key.h>
3. #include <Keypad.h>
4.
5. const byte ROWS = 4;
6. const byte COLS = 4;
7.
8. char hexaKeys[ROWS][COLS] = {
9.   {'1','2','3','A'},
10.  {'4','5','6','B'},
11.  {'7','8','9','C'},
12.  {'*','0','#','D'}
13. };
14.
15. byte rowPins[ROWS] = {9, 8, 7, 6};
16. byte colPins[COLS] = {5, 4, 3, 2};
17.
18. Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
19.                               COLS);
20. const int buzzer = 10;
21.
22. void setup() {
23.   Serial.begin(9600);
24.   pinMode(buzzer, OUTPUT);
25.   noTone(buzzer);
26. }
27.
28. // Takes a character on the keypad and returns a frequency
29. int char_to_freq(char customKey) {
30.   if (customKey == '1') {

```




```

31.     return 524;
32. }
33. else if (customKey == '2') {
34.     return 588;
35. }
36. else if (customKey == '3') {
37.     return 660;
38. }
39. else if (customKey == 'A') {
40.     return 698;
41. }
42. else if (customKey == '4') {
43.     return 784;
44. }
45. else if (customKey == '5') {
46.     return 880;
47. }
48. else if (customKey == '6') {
49.     return 988;
50. }
51. else if (customKey == 'B') {
52.     return 1046;
53. }
54. else if (customKey == '7') {
55.     return 1174;
56. }
57. else if (customKey == '8') {
58.     return 1318;
59. }
60. else if (customKey == '9') {
61.     return 1396;
62. }
63. else if (customKey == 'C') {
64.     return 1568;
65. }
66. else if (customKey == '**') {
67.     return 1760;
68. }
69. else if (customKey == '0') {
70.     return 1976;
71. }
72. else if (customKey == '#') {
73.     return 2094;
74. }
75. else if (customKey == 'D') {
76.     return 2350;
77. }
78. return 0;
79. }
80.
81. void loop() {
82.     char customKey = customKeypad.getKey();
83.     if (customKey) {
84.         int frequency = char_to_freq(customKey);
85.         // Plays frequency on buzzer
86.         tone(buzzer, frequency);
87.         long startTime = millis();
88.         while(millis() - startTime < 500){
89.             // Display wave on plotter
90.             double period = 20.0 / frequency;
91.             double elapsed_time_seconds = (millis() - startTime) / 1000.0;
92.             int period_count = elapsed_time_seconds / period;
93.             Serial.println(period_count % 2);
94.             delay(5);
95.         }
96.     } else {

```



```
97.     noTone(buzzer);  
98.   }  
99. }
```

G. Activity Description (continued)

In this lesson, we use the Keypad library, which makes using the keypad very easy. The line "customKeypad.getKey()" in the loop function returns the current key that is being held down. If there is no key held down, it returns 0.

In our loop function, we get the frequency we want to play using the information about which key is being held down, and we play that frequency for half of a second by calling the "tone" function, with the arguments pin number, frequency, and note duration.

Open the Arduino IDE and upload the code to the Arduino. Open the serial plotter. Try pressing a few buttons on the keypad. As the frequencies go higher, what do the notes sound like? Do they sound higher or lower pitched? Now look at the serial monitor. What happens to the period of waves as the frequencies move higher? Do the waves look closer together, or farther apart?

Look back at the equations, and try to explain this behavior.

H. What Could Go Wrong

- The buzzer may need to be reversed for it to work in order to match the positive/negative charge flow.
- Make sure the frequencies being played by the buzzer are audible. If you think the buzzer is broken, try a pitch that we know can be heard, for example 1000 Hz.

I. Challenge

Reprogram the Arduino to play a different set of frequencies. Test to see what the range of human hearing is by using the Piezo buzzer. Can you hear a note at 10 Hz? Can you hear a note at 5000 Hz?

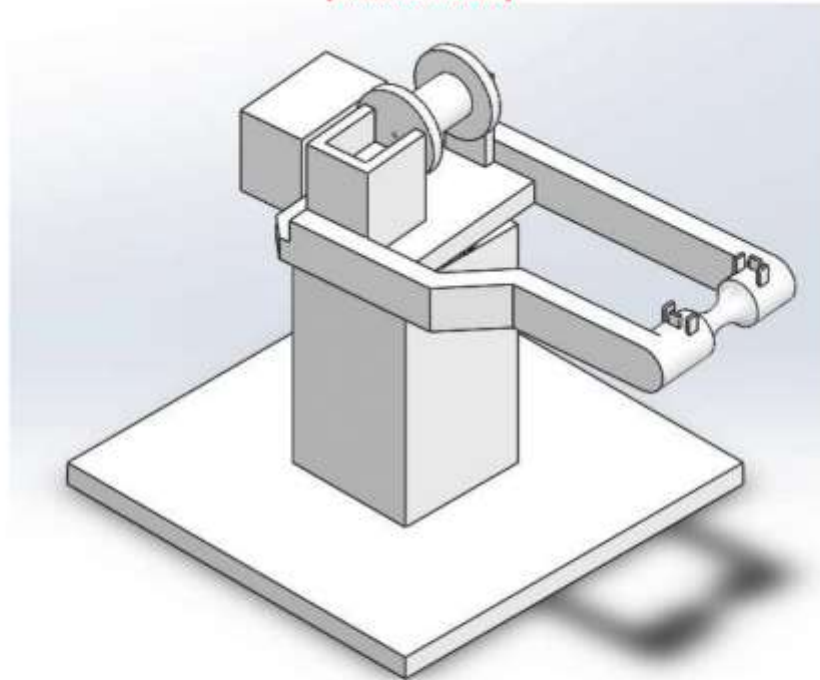
Once you have an estimate of what you think the range of human hearing is, look up online what the actual answer is. Is your answer correct or close to the answer? If your answer is off, why do you think that is? Could it be because of the buzzer?

Appendix F



BUILDING A CRANE

[Arduino STEAM]





A. Title of the Project:

BUILDING A CRANE

B. Subjects

Computers, Physics, Circuitry

C. Knowledge

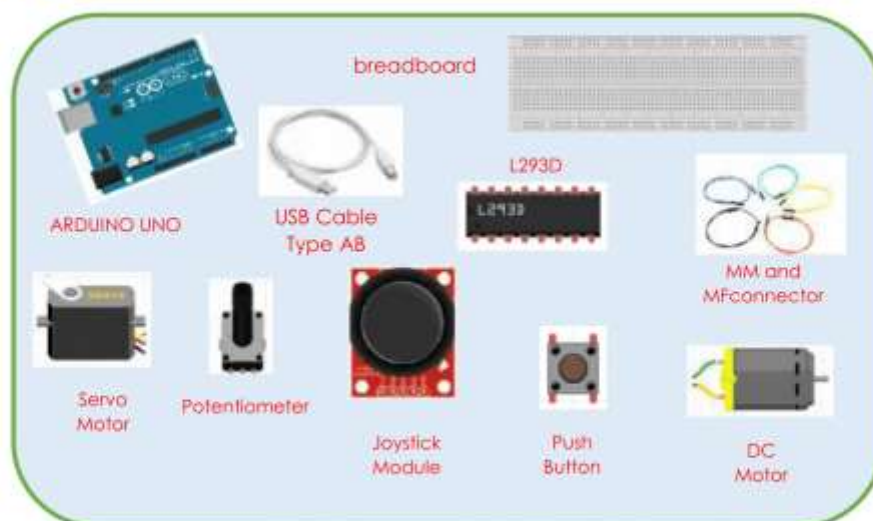
WHAT YOU SHOULD KNOW

- Wiring a circuit
- Using potentiometers
- Using push buttons
- Arduino IDE
- Working with breadboards (photoboard)

What you must learn

- Working with the L293D Driver
- Working joystick module
- Working with Servo motor
- Working with DC motor

D. Materials Needed





E. Abstract

You will make your Arduino the controller of a model crane. The crane will be powered by motors and controlled by a push button, potentiometer, and an analog joystick, while using a L293D driver.

F. Context

The world's cities would not be here today if it wasn't for the crane. The crane is a revolutionary invention that allowed for massive buildings to be constructed. In this module, we will be using the Arduino to power and control a model crane. There are two movements our circuit needs to be able to produce: the first is the movement of the crane itself, and the second is the motion of picking something and putting it down.

The joystick is a useful tool found many places, it can deliver multiple inputs in just one module. The joystick found in your kit is analog. Analog can provide inputs valued between 0 and 1023, ranged 0 to 5 in Volts. Your Arduino has 5 analog inputs (pins A0-A5). This is where you will want to hook up your joystick. The joystick in this project will be used to control the direction of the crane.

The servo motor will carry out the output from the joystick. This motor rotates at precise angles, with a maximum rotation of 180 degrees. It provides slow and controlled movement; which is the why this is the motor of choice for controlling the direction of the crane. There are three wires you need to connect. Two of which are for the power: one connects to a voltage source and the other goes to ground. The third connects to the Arduino to receive output, which in our circuit, will be from the potentiometer and the switch.

The DC motor will control the cranes lifting power. The DC motor spins continuously with high torque (turning power), which is perfect for winding up a spool of string to pick up a load. The motor has two leads; the motor's spin direction depends on the direction of the voltage drop across the leads. Since we want the motor to be able to spin both ways, we need a chip that can quickly makes this complex circuitry happen, which is the L293D motor driver.

The L293D motor driver is a type of H-bridge. A H-bridge is a chip designed to be able switch polarities of voltage drops. It does this with groups of switches around the input and outputs to change the direction of electric current. The one we are using is especially useful for motors, as it allows change the direction of the spin.



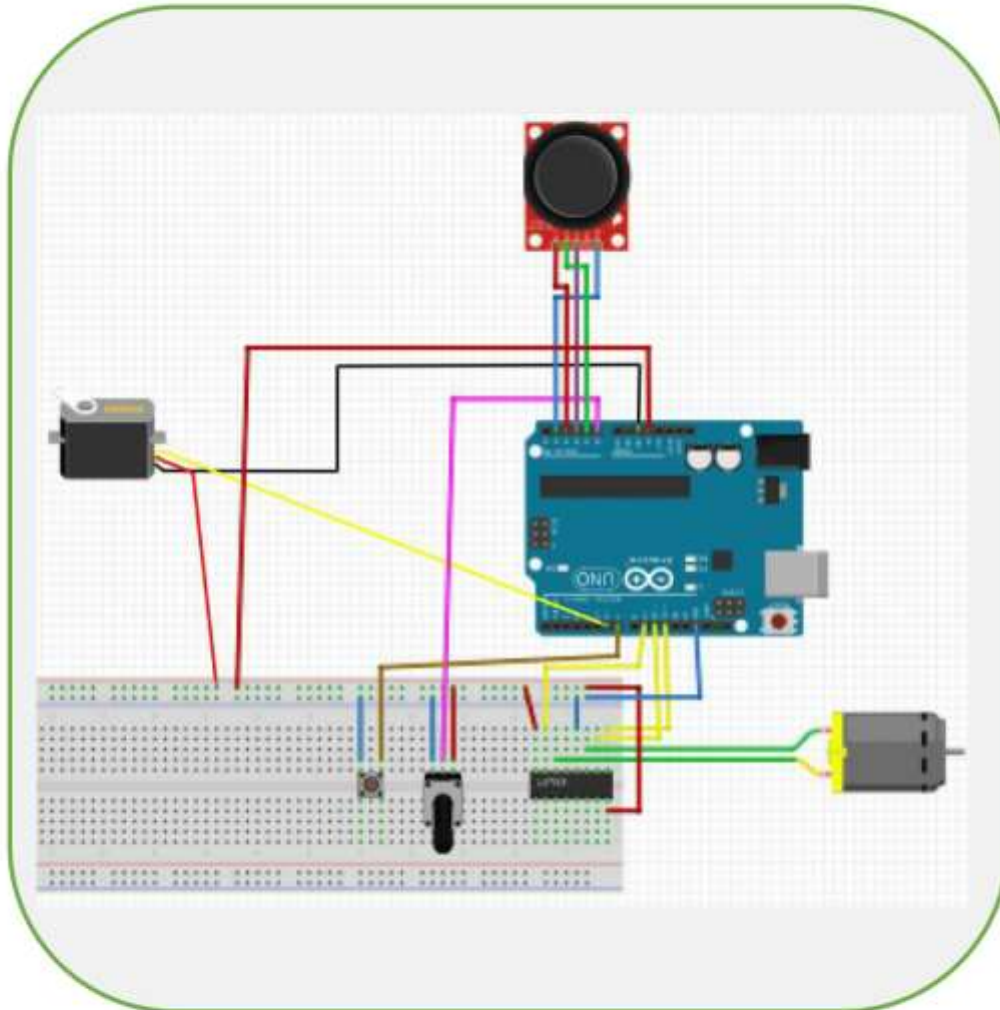


G. Description of Activity

In this workshop you will wire and assemble the model crane. Then you will use it pick up the block and move it to a different spot. The joystick will control the servo motor which points the crane in different directions. To pick something up, turn the potentiometer clockwise to get the DC motor going, which will reel in the string and hook. Then hold down the button to reverse the direction of the spinning to drop something. Once you are done, turn back the potentiometer to stop the crane.



G.1 Connections Diagram





G.2. Code for Programming

```

int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;
int statusPin= 13;

#include <Servo.h>
#define SERVO_PIN 6
#define GROUND_JOY_PIN A4
#define VOUT_JOY_PIN A3
#define XJOY_PIN A2
Servo myservo;
void setup()

{
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(switchPin, INPUT_PULLUP);
  pinMode(statusPin,OUTPUT);

  Serial.begin(9600);
  pinMode(VOUT_JOY_PIN, OUTPUT) ;
  pinMode(GROUND_JOY_PIN, OUTPUT) ;
  digitalWrite(VOUT_JOY_PIN, HIGH) ;
  digitalWrite(GROUND_JOY_PIN,LOW) ;
  myservo.attach(6);
}

void loop()
{
  digitalWrite(13,HIGH);
  int speed = analogRead(potPin) / 4;
  boolean reverse = digitalRead(switchPin);
  setMotor(speed, reverse);
}

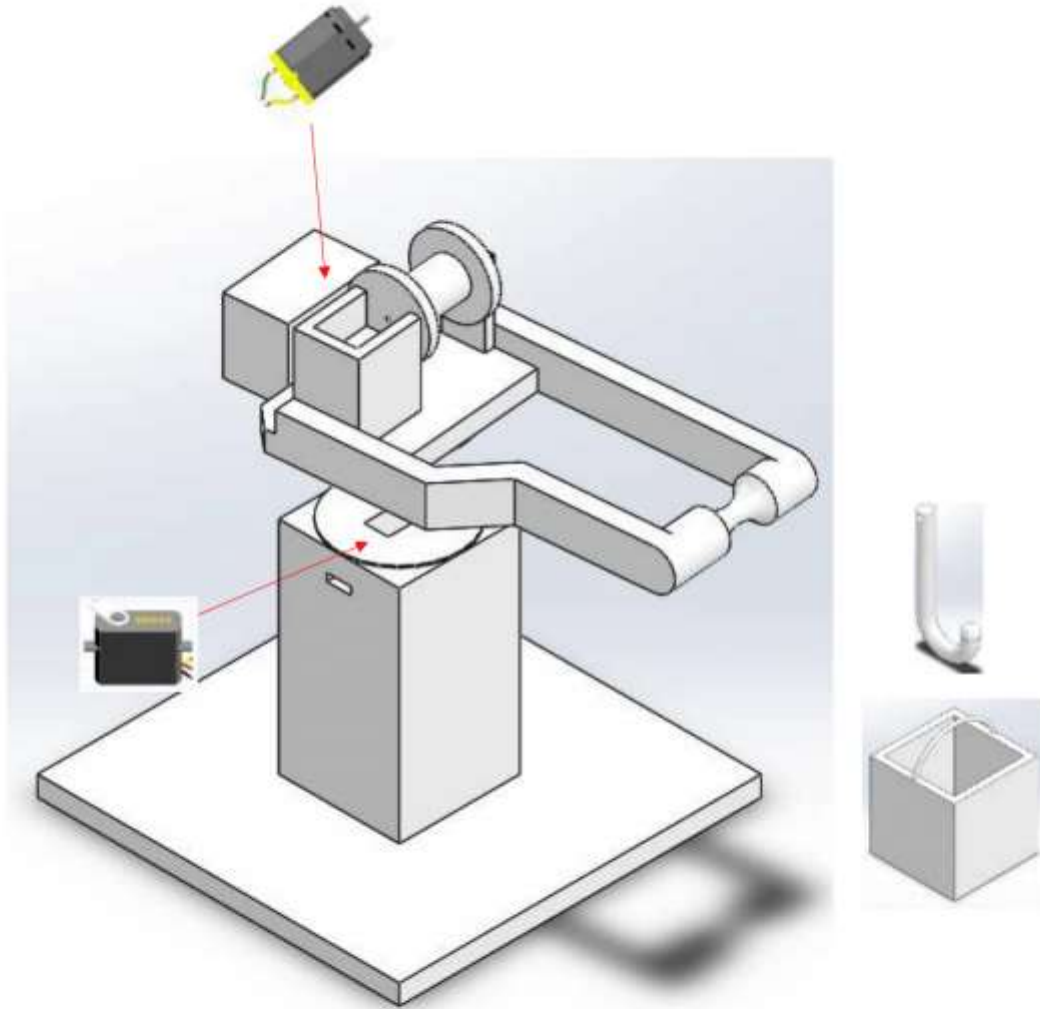
void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);

  delay(200);
  int joystickXVal = analogRead(XJOY_PIN) ;
  Serial.print(joystickXVal);
  Serial.println(" = input from joystick");
  Serial.print((joystickXVal+520)/10);
  Serial.println(" = output to servo");
  Serial.println() ;
  myservo.write((joystickXVal+520)/10);
}

```




G.3 Assembling the Crane





H. What Could Go Wrong

1. Wiring the motor driver incorrectly: Make sure to wire that carefully, there are 16 pins on this chip all very close together.
2. Wiring the joystick incorrectly: This device has three pins all for inputs, double check the pins are matching up correctly to the Arduino.
3. Poor connection on the DC motor: The DC motor's female lead are poor and don't hold wires well, try curving the wire pin around the motor's leads to make it more secure.

I. CHALLENGE

Create code and rewire the circuit, so that you control the whole crane with just the joystick.



H. What Could Go Wrong

1. Wiring the motor driver incorrectly: Make sure to wire that carefully, there are 16 pins on this chip all very close together.
2. Wiring the joystick incorrectly: This device has three pins all for inputs, double check the pins are matching up correctly to the Arduino.
3. Poor connection on the DC motor: The DC motor's female lead are poor and don't hold wires well, try curving the wire pin around the motor's leads to make it more secure.
- 4) The buzzer might not sound, and it is most likely it is wired backwards, just flip around the pins and it should work.
- 5) You are using a large number of adjacent digital pins, make sure all your wires are pushed in all the way.
- 6) The RGB LED has four pins that are close together, plugging it in the breadboard can be difficult, make sure all the pins are in all the way.

I. Challenge

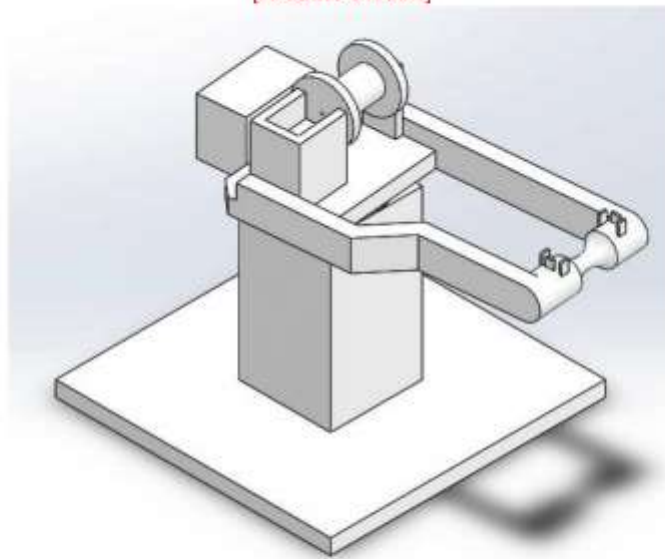
You are tasked in making the crane even safer and more accident proof. What features can you add? Be creative! If you can't think of anything, try recoding so that the DC Motor shuts off automatically when the detector reads a really small distance.

Appendix G



IMPLEMENTING SAFETY FEATURES

[Arduino STEAM]





A. Title of the project:

IMPLEMENTING SAFETY FEATURES

B. Subjects

Computers, Physics, Circuitry

C. Knowledge

WHAT YOU SHOULD KNOW

- Wiring a circuit
- Using potentiometers
- Using push buttons
- Arduino IDE
- Working with breadboards (photoboard)
- Working with L293D Driver
- Working with joystick module
- Working with Servo motor
- Working with DC motor
- Working with Ultrasonic Detector

What you must learn

- Working with the Active Buzzer
- Working RGB LED
- Importance of Safety Measures in Engineering



D. Materials Needed



E. Abstract

You will continue the crane from the last module. This workshop is focused on improving the crane by adding safety features. You will learn about the importance of safety features in engineering and how they are implemented.

F. Context

One aspect of engineering that is very important is making sure the machines we use are safe. Limiting potential accidents saves people from injury and death. Safety applications is a growing and advancing topic within engineering. Example of this is some new cars today have automatic braking features if the car detects an object is very close. A more familiar example would be your car beeping when a seatbelt is unfastened.

Our goal is to implement safety features that will limit poor control of the crane. One possible mistake when operating the crane is letting the DC motor pick up an object too far so that it collides with the crane. First thing we need is to



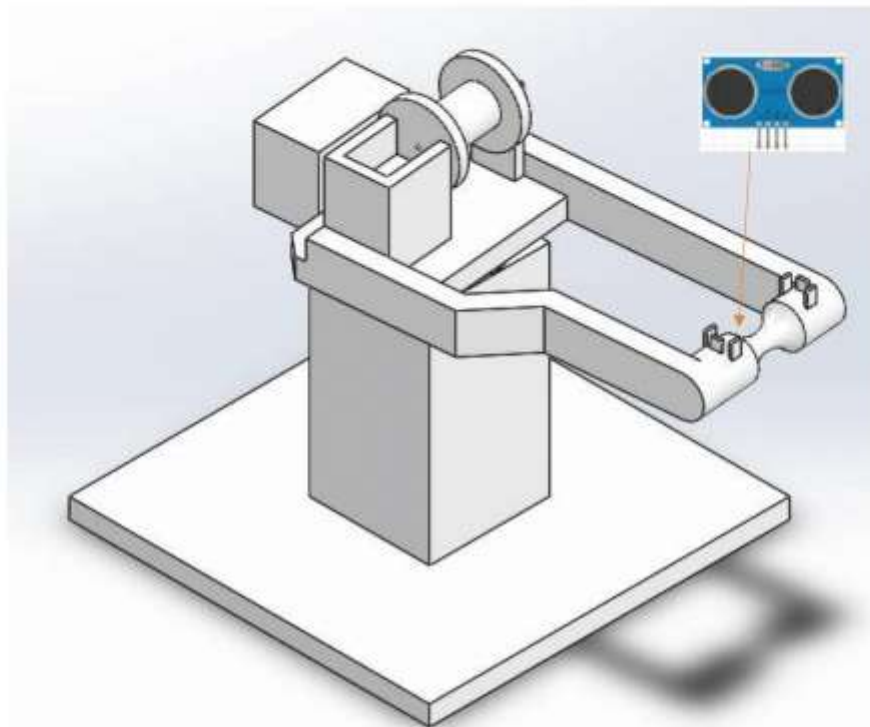
implement a way for a circuit to detect a possible hazard. We can use the ultrasonic detector to do this. If the detector is on top of the crane pointed down, it can tell the Arduino how far away the object that is being picked up is from the crane arm. This is useless if there's not another component that will act depending on the distance the detector records, otherwise the detector doesn't fix any problems. So, we will implement a buzzer and a RGB LED to the circuit.

The active buzzer has two pins and easy to set up. One pin will be connected to the ground, and the other will be connected to a digital pin, which will be an output pin. To code it to make a noise, use **tone(int, 1000)**. The int will defined beforehand as 'buzzer pin'. The number is how many hertz the noise's frequency will be. To shut off the buzzer use **noTone(int)**. With if statements, we will make the Arduino make the buzzer go off when the detector returns a distance that is really close. This will alert the user that a collision is coming soon, so they should stop the DC Motor.

The RGB LED is a useful part in your kit. It can emit a variety of colored light all from the same bulb. We want this LED to emit green when the detector reads safe distances, and then turn red when the object starts getting to close to the crane arm. Along with the buzzer, it will alert the controller of possible problem. The RGB LED works by being able to emit the three primary colors, then mixing them to produce other colors, or just plain white light. The LED has four pins, one pin for each primary color and one that connects to the ground. The color pins will be connected to the digital pins on the Arduino and will be output pins. The LED uses Pulse Width Modulation (PWM). In a nutshell, it is similar to how analog works, but through the digital pins. It gives off a steady signal between 0 and 5 volts. PWM has values between 0 and 255. So, to have the LED flash just green, you want to code the green pin at 255, and other color pins and 0. You can see how that works by looking at the code where the setColor functions are.

G. Description of Activity

In this workshop you will reassemble the crane from the last workshop. When you pick something up with the crane, slowly continue to rise the item until you set off the LED and the buzzer, then lower the object until they shut off.

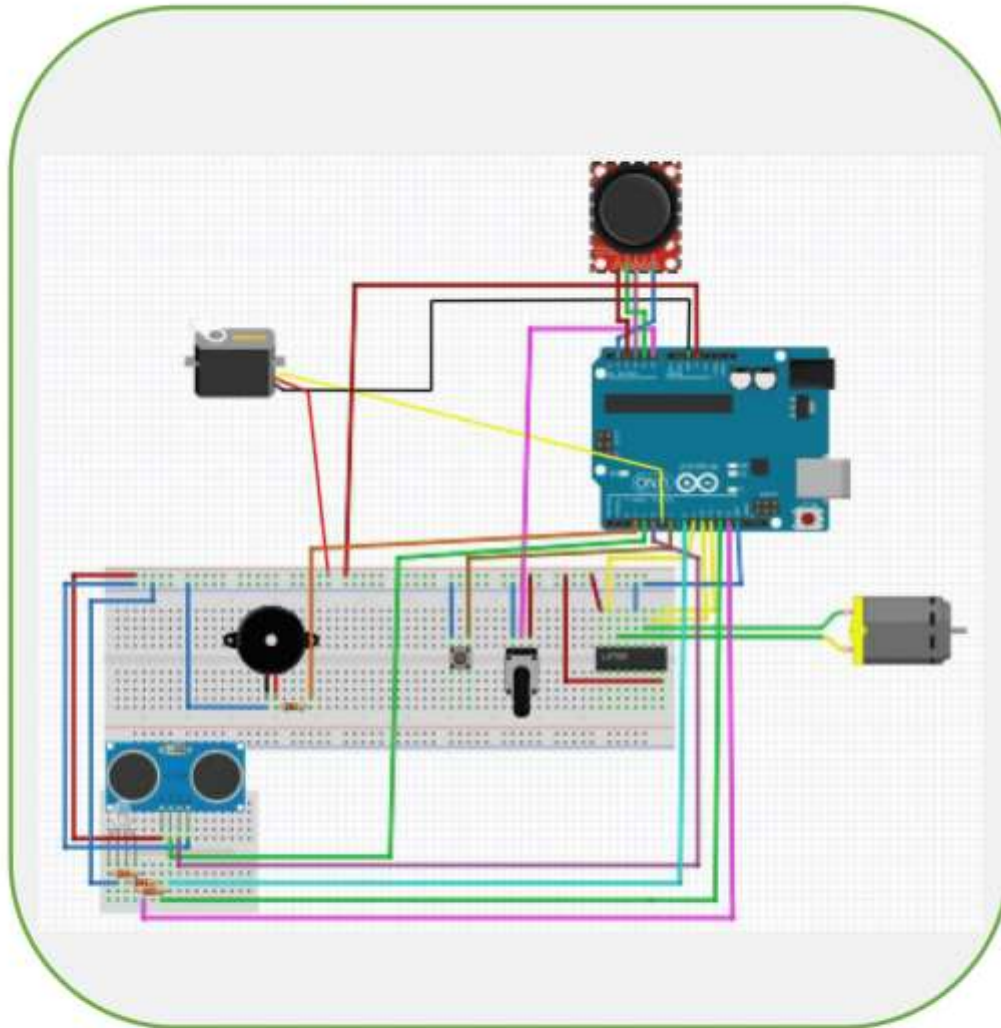


To get the ultrasonic detector to fit in place, you need to put it on the mini breadboard separate from the rest of the circuit. Then the mini breadboard will slide in the notches and be steady. Set up the wiring the detector and LED in similar fashion as the example. Also, it is important that you use your longest wires to connect this. These parts will be on the crane and relatively far away from the Arduino, so it is imperative that these parts are reasonably mobile and free.





G.1. Connections Diagram





G.2. Code for Programming

```

int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;
int statusPin= 13;
const int trigPin = 4;
const int echoPin = 5;
const int buzzer = 3;
int redPin = 13;
int greenPin = 12;
int bluePin = 8;

long duration;
int distance;

#include <Servo.h>
#define SERVO_PIN 6
#define GROUND_JOY_PIN A4
#define VOUT_JOY_PIN A3
#define XJOY_PIN A2
Servo myservo ;
void setup()

{
pinMode(in1Pin, OUTPUT);
pinMode(in2Pin, OUTPUT);
pinMode(enablePin, OUTPUT);
pinMode(switchPin, INPUT_PULLUP);
pinMode(statusPin,OUTPUT);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(redPin, OUTPUT);
pinMode(greenPin, OUTPUT);

Serial.begin(9600);
pinMode(VOUT_JOY_PIN, OUTPUT) ;
pinMode(GROUND_JOY_PIN, OUTPUT) ;
digitalWrite(VOUT_JOY_PIN, HIGH) ;
digitalWrite(GROUND_JOY_PIN,LOW) ;
myservo.attach(6);
}

void loop()
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

```



```

digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration * (0.034 / 2);

Serial.print("Distance: ");
Serial.println(distance);

digitalWrite(13,HIGH);
int speed = analogRead(potPin) / 4;
boolean reverse = digitalRead(switchPin);
setMotor(speed, reverse);
}

void setMotor(int speed, boolean reverse)
{
analogWrite(enablePin, speed);
digitalWrite(in1Pin, ! reverse);
digitalWrite(in2Pin, reverse);

delay(200);
int joystickXVal = analogRead(XJOY_PIN) ;
Serial.print(joystickXVal);
Serial.println(" = input from joystick");
Serial.print((joystickXVal+520)/10);
Serial.println(" = output to servo");
Serial.println() ;
myservo.write((joystickXVal+520)/10);

if (distance < 9)
{
tone(buzzer, 1000);
setColor(255,0,0);
delay(2000);
}
else (distance >9);
{
noTone(buzzer);
setColor(0, 255, 0);
}
}

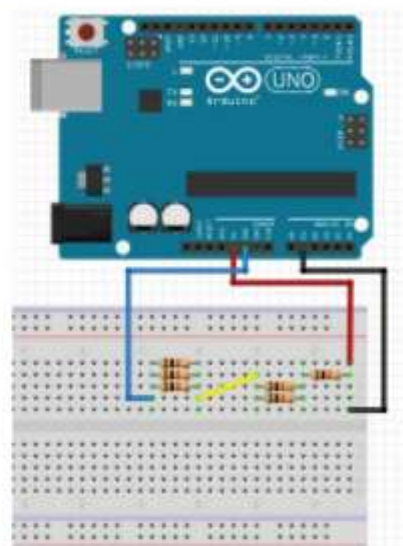
void setColor(int redValue, int greenValue, int blueValue) {
analogWrite(redPin, redValue);
analogWrite(greenPin, greenValue);
analogWrite(bluePin, blueValue);
}

```

Appendix H



INTRODUCTION TO ELECTRICITY



[Arduino STEAM]





A. Title of the Project:

INTRODUCTION TO ELECTRICITY

B. Subjects

Electricity and Math

C. Knowledge

Prior Knowledge

- Familiarity with the Arduino IDE
- Resistors
- Wiring with breadboards.

What You Should Learn

- Introductory Concepts of electricity: What it is, Coulomb's Law, Ohm's Law, Resistor Addition
- Voltmeters
- Circuit Analysis

D. Materials Needed



E. Abstract

In this workshop you will set up a circuit that can measure voltages and you will use it analyze circuits. You will learn the basic theories and laws behind electricity and prove them with an experiment.



F. Context

Electricity is one of the most important natural phenomena in the world. In modern society, you will find electricity and its applications almost everywhere, your Arduino, your phone, your computer, or just the lights in the room. It allows us to technologically advance quickly and helps us in our everyday life.

So, what is electricity exactly? It is a flow of electric charge. It is the structure of the atom that makes it possible for charge to flow. An atom has a core in the center composed of protons and neutrons, called the nucleus; and electrons orbit the nucleus. We want to focus on the valence electrons, these are the electrons that are in the outer most orbit, called the valence shell. These electrons are the most loosely held electrons, which makes them able to move around between atoms under certain circumstances.

Charge is carried by atoms. Protons have a positive charge, neutrons do not carry a charge, and electrons carry a negative charge. The charge of an electron is equal to that of a proton; only difference is the sign of the charge. How charges interact with each other is important, the rule of thumb is likes repel and opposites attract. So, a proton and an electron will be attracted to each other, and two electrons or two protons will repel each other. A charge has the unit of a Coulomb, which comes from the name of Charles de Coulomb, who discovered the law of electrostatic force. This law states that the push/pull force between two charges depends on the magnitude of the charges, and the distance between them. These factors are mathematically related by the following equation,

$$F = \frac{k(q1)(q2)}{r^2}$$

F is the force, the q's are the charges, r is the distance, and k is just a constant. With electrostatic force between valence electrons in a group of atoms, electric flow is possible. Electricity is a chain reaction, visualize a free electron coming into an atom, and taking the spot of one the electrons, kicking one out. That electron will then go on to the next atom and take the place of another electron. This will continue and now we have a flow of charge. This only happens when the electrons have a fully complete and closed loop to flow through. This loop has to be composed of material that has valence electrons that move easily. These materials are called conductors and can be found in the 11th group of the periodic table, some common examples are copper and gold.

What about the concept of flowing charges gives it the ability to power components such as lights or motors? To help explain this, we need to go over the electric field. Picture a group of positive charges, then some distance from that is a group of negative charges. In between the groups is a single positive charge, next to the other positive charges. What would happen if it were released? It would accelerate towards the negative charges. So, before it





moves and gains kinetic energy, it has electric potential energy. In a circuit, the electricity has potential energy, and when it flows through a component such as an LED, its potential energy is converted into a different form of energy, such as light.

The way to measure the amount of potential energy is through the term called voltage. However, voltage doesn't directly translate to electric potential energy; it describes *electric potential*, which is the amount of potential energy per charge. That is described by the units of Joules per Coulomb, or a volt(V).

There is another term that is used to describe electricity. This is called the ampere(amp). This refers to the current, which is the rate of charges flowing past a certain point over a period of time. Amps(I) are equal to Coulombs per second.


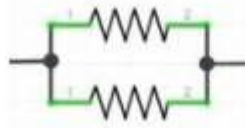
There is important equation that relates volts and amps, it is the stepping stone to beginning to do circuit analysis. It comes from Ohm's Law. It states that the voltage is proportional to the current across a conductor. The constant of proportionality is the resistance (R). Resistance is in the unit Ohms(Ω). The more Ohms, the more the current through the circuit is impeded, and the bigger the drop in the voltage. Voltage, current, and resistance are mathematically related by the following formula.

$$V = IR$$

Now that we have an understanding of resistance, we can look at the role that resistors play. Why would we want something that slows down current and causes voltage drops? It's because it keeps you and the circuit components safe. Certain things can only handle so much current going through them. If they do, they can melt or catch fire, which will damage the circuit, and more significantly, cause a great danger to the people around it.

When you implement resistors into a circuit, there is two different ways to implement them. You can add resistors in series, or in parallel. When in series, the resistance grows in the circuit. However, when in parallel, the more you add, the less the total resistance is, because you are opening up more paths for current to flow. The total resistance is called the equivalent resistance (R_{eq}). Key thing to note about series and parallel is that current is the same for both parts in series, and the voltage is split; and in parallel the current is split and the voltage is the same for the parts.



Series	Parallel
	
$R_{eq} = R_1 + R_2 + \dots + R_n$	$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n}$

G. Description of Activity

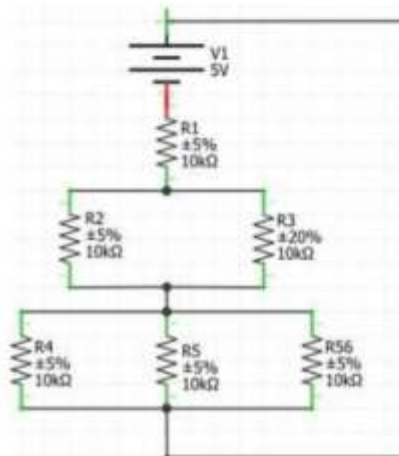
The circuit you are going to build will be able to measure the voltage drops across the resistors. The first thing we will do though, is to calculate the voltage drops, then we will measure them. The experiment should prove the theory.

Looking at the circuit diagram, you can see that you can break the circuit into three stages of resistors: the first one is just one resistor, the second is two resistors in parallel, then the last is the three resistors in parallel. Each 'stage' is in series.

The first step is to calculate the resistance in each stage. The first is just 10k Ohms, the second one is 5k Ohms. Using the parallel resistance equation, calculate the third stage. Then lastly add in series the three stages to find the equivalent resistance.

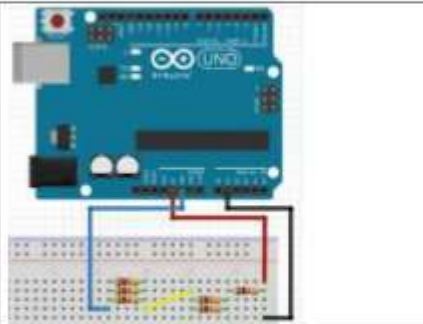
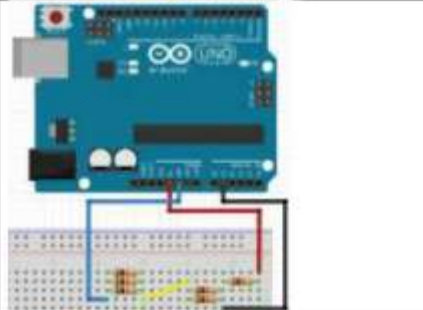
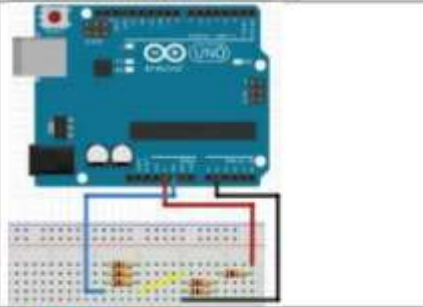
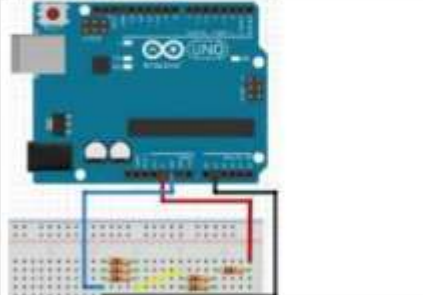
The next step is to use the R_{eq} to find the current going through each stage. You can do this with Ohm's law by acquiring the equation: $I = \frac{V}{R_{eq}}$. We are using the 5V pin on the Arduino, so use 5V and the R_{eq} you calculated to find the current.

The last step is to calculate the voltage drop at each stage. Again, we will use Ohm's Law: $V = IR$. Use the current you calculated in the last step, and the resistance at each stage, so this will be three calculations.





Now we will use the circuit to find these voltage drops. The wire that is connected to A1 is the voltmeter. To find a voltage drop, you need to record the value at top of stage, and the value at the bottom of the stage, then subtract the values.

Voltmeter Recording 1	
Voltmeter Recording 2	
Voltmeter Recording 3	
Voltmeter Recording 4	



How close are your experimental values to your theoretical values? They should be very close. A tool used to calculate the accuracy of the experiment is the percent error equation. This puts a numerical value to show how accurate your findings are. It is calculated by the simple equation below.

$$\text{Percent Error} = \frac{|Value(\text{experimental}) - Value(\text{theoretical})|}{Value(\text{theoretical})} * (100)$$

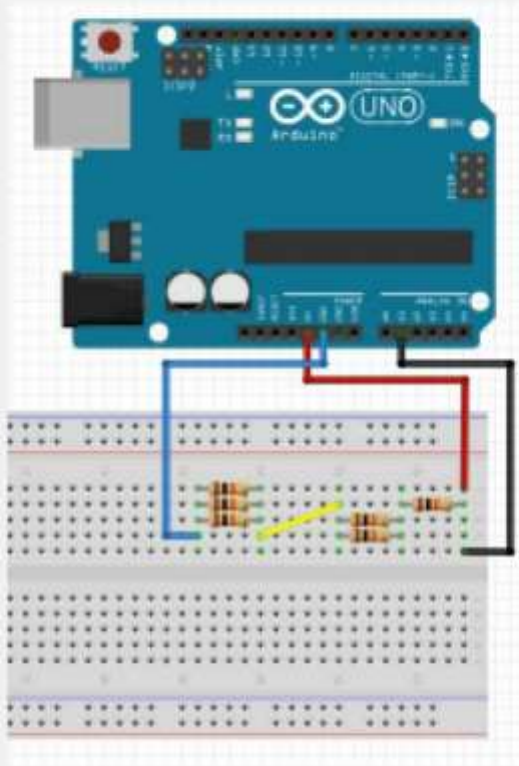
A good goal to aim for is to have your percent error at 5% or less. If it is bigger, you may have made a mistake in either your calculations or in your experiment.

Fill in the table with your findings.

Stage	Theoretical Value	Experimental Value	Percent Error
1			
2			
3			



G.1. Connections Diagram





G.2. Code for Programming

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  int sensorValue = analogRead(A1);  
  float voltage = sensorValue * (5.0 / 1023.0);  
  Serial.print("Voltage: ");  
  Serial.println(voltage);  
  delay(500);  
}
```

H. WHAT COULD GO WRONG?

You are using a lot of resistors in this circuit, make sure you have the right ones, and correctly lined up on the breadboard.

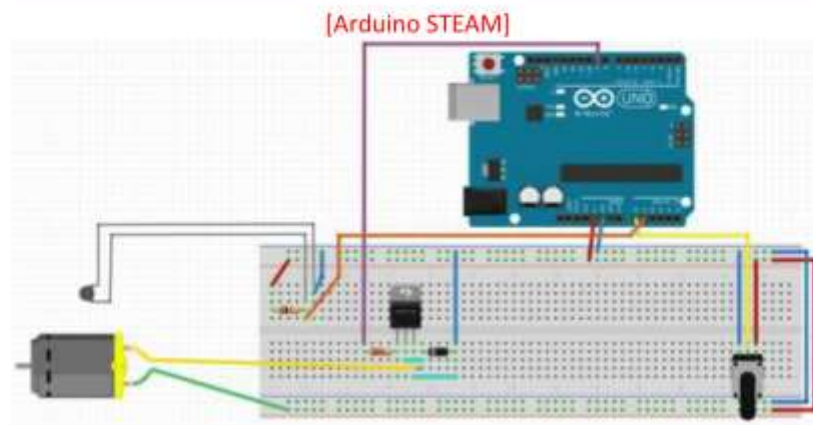
I. CHALLENGE

How can you achieve the same equivalent resistance that we had in this workshop only using 100K Ohm resistors. Calculate the current and voltage drop of the resistors, and then use circuit to double check your work.

Appendix I



WORKING WITH THERMISTORS





A. Title of the project:

WORKING WITH THERMISTORS

B. Subjects

Computers, Math, Circuitry

C. Knowledge

WHAT YOU SHOULD KNOW

- Wiring a circuit
- Using potentiometers
- Arduino IDE
- Working with breadboards (photoboard)
- Working with DC motor
- Using resistors

What you must learn

- Working with transistors
- Working with diodes
- Working with thermistors



D. Materials Needed



E. Abstract

In this workshop you will implement a very common and safety feature by setting up a circuit that will shut off a motor when it gets too hot. You will learn about temperature regulation and its importance. You will also learn about how it is implemented into machines and other devices.

F. Context

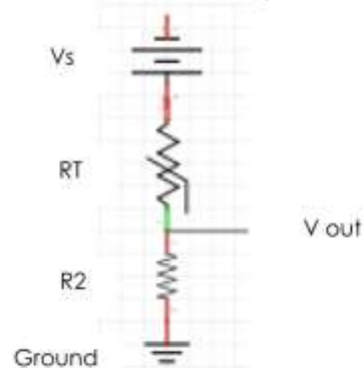
Thermistors and other types of temperature sensors are an important tool within engineering, and they have vast applications. They keep machinery and people safe. In a lot of machinery, overheating can be a potential significant problem, and thermistors can control that problem. In circuits, energy surges are a common problem; thermistors solve this by being used in surge protectors/power strips. Thermistors can also be found in your car. They are used to alert the driver if certain parts of the engine are running too hot. Then the driver can get the car fixed before it breaks down. Rechargeable batteries are made possible by thermistors. When a charging process is underway, overheating tends to happen



easily. Temperature regulation is small but a very valuable feature that is implemented across a variety of machines.

So how do thermistors work? The word comes from a combination of thermometer and resistor. It is made with a bead of semiconducting material with two leads attached to it. The material has the property that its resistance changes with the temperature of the material. There are two types of thermistors, negative temperature coefficient (NTC), and positive temperature coefficient (PTC). NTC is far more common. NTC means that as temperature increases, the resistance decreases; and PTC means that as temperature increases, so does the resistance.

The thermistor acts as an input on the Arduino. The voltage drop across the thermistor is specific to a certain temperature. For the Arduino to read the voltage drop, there needs to be a voltage divider set up in our circuit. A voltage divider is a simple set up that splits up the voltage source. The reason we need this is because if we just have the thermistor, the volt reading will be either be just equal the voltage source, or if the input pin is on the other side of the thermistor, it will just read zero. If we to have another resistor in series, so we can read the changes in resistance of the thermistor. This setup will look like this:



There is equation for this specific setup called the voltage divider equation.

$$V_{out} = V_s * \frac{R_2}{R_T + R_2}$$

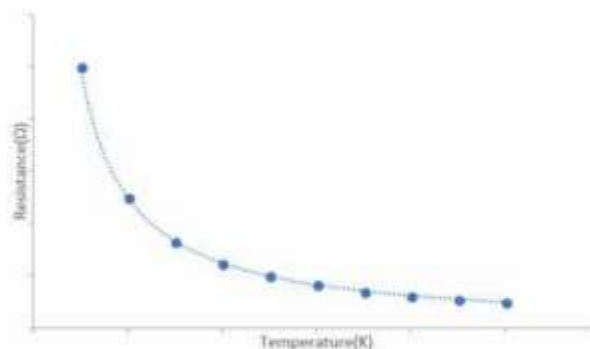
Temperature is more directly related to the resistance of the thermistor, so we will algebraically rearrange the equation to obtain the following equation.

$$R_T = R_2 * \left(\frac{V_s}{V_{out}} - 1 \right)$$



We will have this equation incorporated into our code,

We are not done yet understanding how to obtain a temperature reading from the thermistor. The relationship



between a thermistor's resistance and temperature is not linear. Therefore, there is more math to be done to get there. Every thermistor has a "beta" value, this number dictates the specific shape of the curve of the graph for the thermistor. Your beta value has been given to you in the code, so you won't have to calculate it. However, the equation that relates beta, resistance, and temperature is as follows:

$$\beta = \frac{\ln\left(\frac{R1}{R2}\right)}{\left(\frac{1}{T1}\right) - \left(\frac{1}{T2}\right)}$$

R1 and T1, and then R2 and T2 are data points from anywhere on the curve. So, we have the beta value, the Arduino reads both resistance values, and T1 is usually taken to be room temperature, which leaves only the variable T2, which is what the temperature your Arduino, thermistor, and code will calculate.

The next part that this workshop introduces is a transistor. Invented by William Shockley, Walter Brattain, and John Bardeen, transistors are one of most significant technological invention in the 20th century. Since their invention, computers, phones and other electronics to significantly advance. Transistors can operate to do two main functions. They can either act as an amplifiers or switches. They are cheap, lightweight, efficient, and durable.

The last new part that is used is the diode. Diodes only allow current in one direction. In our circuit we have it connected to our transistor. This is because the transistor is acting as an amplifier, and this diode keeps the amplification in check by imposing a limit on how much the voltage can be amplified.

G. Description of Activity

In this workshop, you will set a circuit that will protect the DC motor. Your job is to keep the motor safe from overheating so that the part is not damaged and limit possible fire hazards.



Make sure you have the thermistor connected into female wires, rather than directly in the breadboard, so it is easy to have thermistor against the motor. Lie the thermistor flat along the surface of the motor, the more surface area in contact with the motor, the quicker the thermistor will heat up to the motor's temperature. Ideally you should use a small piece of tape to hold it down on the motor. Because your hands will heat up the thermistor and you will get skewed temperature readings. If you do not have tape, you can place the thermistor under the motor against the table, and lightly press the motor down on top of it to keep it in place.

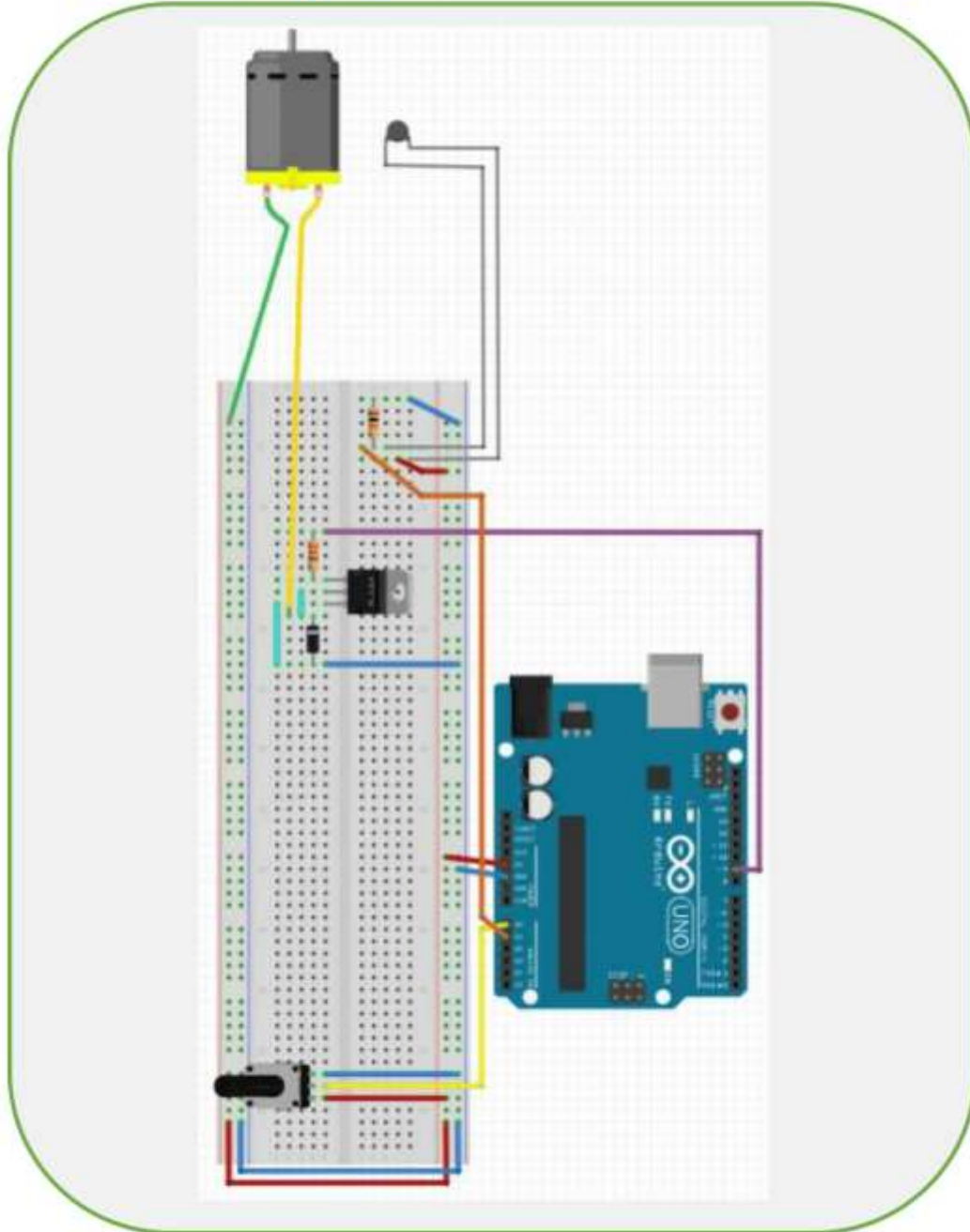


Once it is all set up, upload the code to the Arduino board. Open up the Serial Monitor to see the temperature readings to make sure the thermistor is operating correctly. Use the potentiometer to slowly increase the motor's speed. As you do this it should start to get hotter. If the motor at all makes the thermistor read 30 degrees Celsius, the motor should automatically shutoff, until the temperature returns to back under 30 degrees. If for whatever reason your motor never gets that hot, use your fingers and squeeze the bead on the thermistor to warm it up to see the auto shut off feature in action.

You have recreated one of the most widely used and important safety features in engineering!



G.1. Connections Diagram





G.2. Code for Programming

```
int potPin = A0;
int sensorValue = 0;
int outputValue = 0;
int transistorPin = 9;

#define RT0 10000
#define B 3860
#define VS 5
#define R 10000

float RT, ln, TX, T0, VRT;

void setup()
{
  Serial.begin(9600);
  pinMode(transistorPin, OUTPUT);
  T0 = 25 + 273.15;
}

void loop()
{
  VRT = analogRead(A1);
  VRT = (5.00 / 1023.00) * VRT;
  RT = R * ((VS/VRT) - 1);

  ln = log(RT / RT0);
  TX = (1 / ((ln / B) + (1 / T0)));
  TX = TX - 273.15;

  Serial.print("Temperature:");
  Serial.print("\t");
  Serial.print(TX);
  Serial.println("C\t\t");
  delay(500);

  sensorValue = analogRead(potPin)/4;
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(transistorPin, sensorValue);

  if (TX > 30)
  {
    analogWrite(transistorPin, 0);
  }
}
```



H. What Could Go Wrong

- 1) The wires connected to the DC motor are loose, causing poor connection. Make sure the wires are firmly pressed against the leads.
- 2) The thermistor isn't pressed into the female wires enough, if this is the case, you will notice the temperature reading will be 0 degrees.
- 3) The resistors could potentially be swapped. If this is so the motor will be going slow and your temperature readings will be completely off.
- 4) The diode is backwards. There is a gray stripe on one side of the diode. Make sure it is facing the same direction as it is in the diagram.
- 5) The transistor is flipped the wrong way, easy fix!

I. Challenge

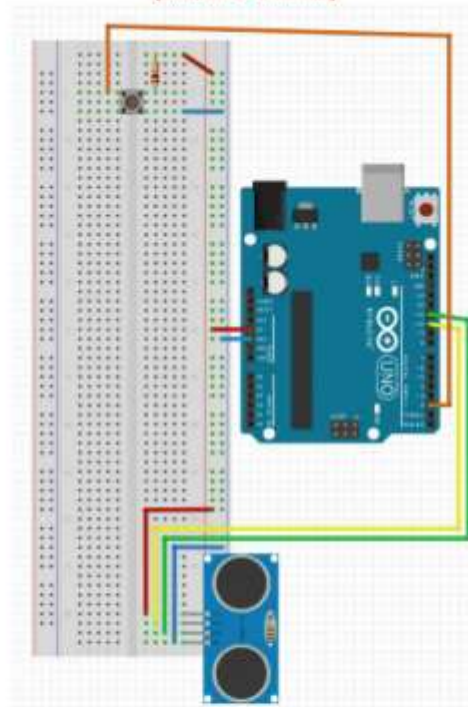
You have learned another important safety feature. Can you go back and apply this to the crane safety workshop?

Appendix J



ULTRASONIC DETECTION

[Arduino STEAM]





A. Title of the Project:
Displacement, Velocity, Acceleration

B. Subjects
 Math, Physics, and Computers

C. Knowledge

WHAT YOU SHOULD KNOW

- Familiarity with Microsoft Excel
- Familiarity with the Arduino IDE
- Resistors
- Wiring with breadboards

WHAT YOU MUST LEARN

- Working with the ultrasonic detector
- Working with a push button
- Concepts of displacement, velocity, and acceleration in physics
- Data management and graphing in Microsoft Excel

D. Materials Needed





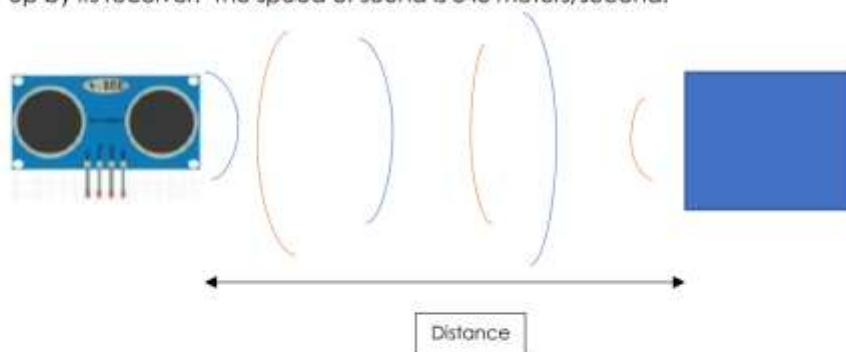
E. Abstract

In this workshop you will set up a circuit that can detect and record the distance of a rolling plastic bottle. This lesson and experiment will teach about basic kinematics of displacement, velocity, and acceleration.

F. Context

Sonic and echo location is used more than you might realize. There are several kinds of animals that use it to travel, such as bats and dolphins. Human use this technology for a variety of different reasons. It is used for navigation, medical purposes, and engineering. Ships and submarines used sonar to avoid collisions. The medical industry used ultrasound to safely scan bodies to detect possible problems. In engineering, we can perform nondestructive testing in which we can scan machines for cracks or other problems, without having to take it apart.

The ultrasonic detector works by one its parts emits a high-pitched frequency, and then it times how long it takes for its sound to come back and be picked up by its receiver. The speed of sound is 343 meters/second.



The equation for distance is: $d = v * t$, where v is the speed of sound, and t is time in seconds, which is what our device records, however, the detector times how it long it takes for the sound wave to reach an object and come all the way back. So, in this case our equation will be: $d = \frac{(v*t)}{2}$. For example, if the detector records 2 seconds to receive its ultrasound signal, that means the object is $\frac{343\text{meters}}{\text{seconds}} * (2\text{seconds})}{2} = 343$ meters away.



The ultrasonic detector has 4 pins to connect. The ones on the outside are for power and ground, so those should be connected to the power rails on the breadboard. The pins on the inside will be connected to digital pins on the Arduino. The first one is called "trig." This is an output pin, the tells the detector to send ping of ultrasonic sound. The next pin is called "echo," this is an input pin; this pin will receive and read the echo from the ping sent from trig pin.

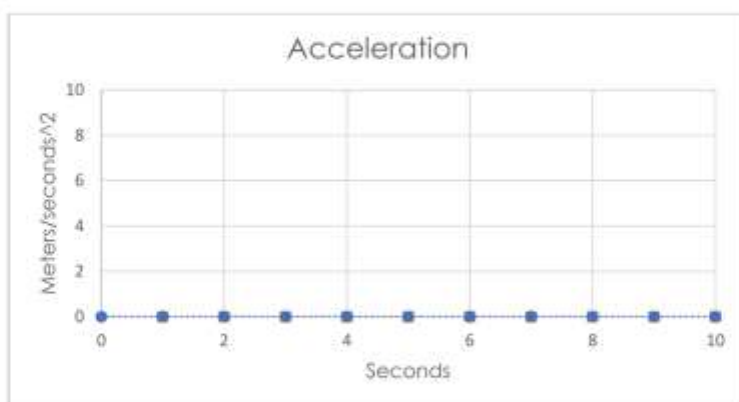
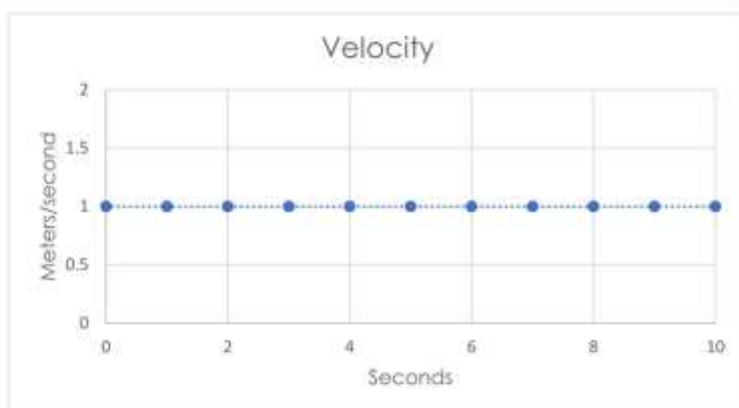
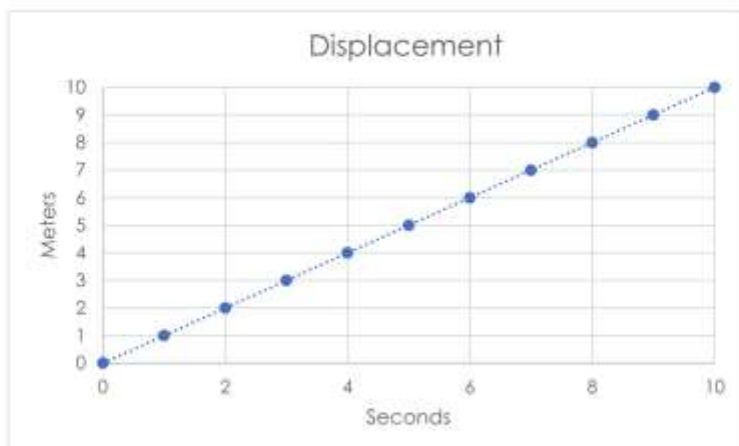
When the ultrasonic detector is turned on, it will record data continuously, which is not ideal because we want recordings for a precise period of time. Therefore, we need to implement a feature that will turn on and off data collection, so we will utilize the push button. This way the data we have will only be the data that we want.

The push button acts like a momentary switch, when it is pushed down, it will complete a circuit, and when it is released it will break the circuit. To one pin, you should connect the power source, and in the adjacent you should connect the ground. In the loop, you should implement a resistor, a 10k Ohm will be sufficient. **Be sure to never use a low-ohm resistor in place of high-ohm resistor.** It is better to use too big of a resistor, than too small of one. You don't want a part receiving more current than it can handle. In the pin opposite to the V_{CC} pin, connect to a Arduino digital pin. This will be the input—it gives a signal to the Arduino.

So, what are the concepts of displacement, velocity, and acceleration; and how are they related? Displacement is described the net change of position of an object. The velocity is the rate of change of an object's displacement. For example, if an object travels 8 meters over a time period of 2 seconds, then its velocity was 4 meters per second. Lastly, acceleration is the rate of change of velocity. If at an instant our object is traveling at a velocity of 2 meters per second, and then 3 seconds later, it's traveling at 8 meters per second, then its velocity changed by 6 meters per second over the course of 3 seconds, that means the object was accelerating at a rate of 2 meters per seconds squared,



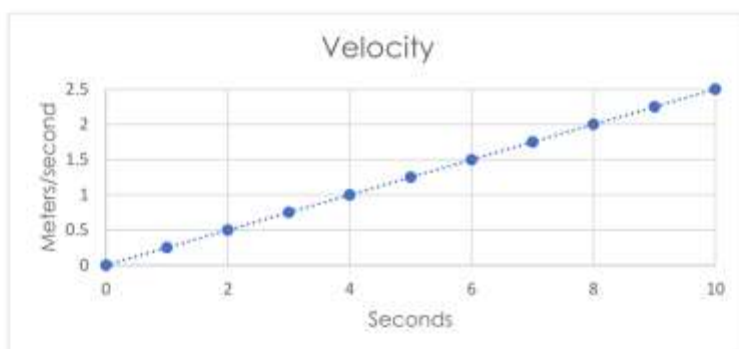
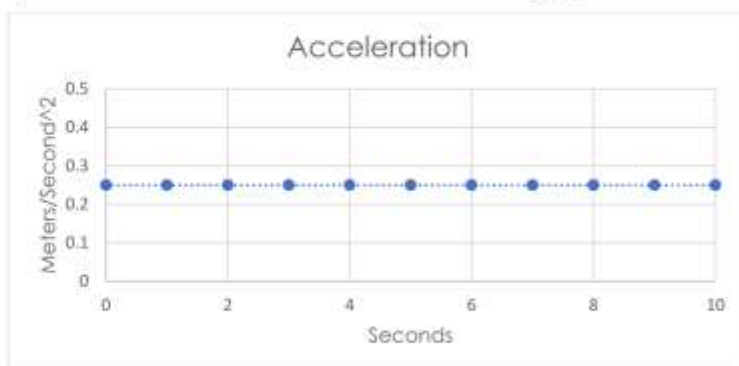
Let's say a ball rolls and we record its motion, and come up with the following graphs:



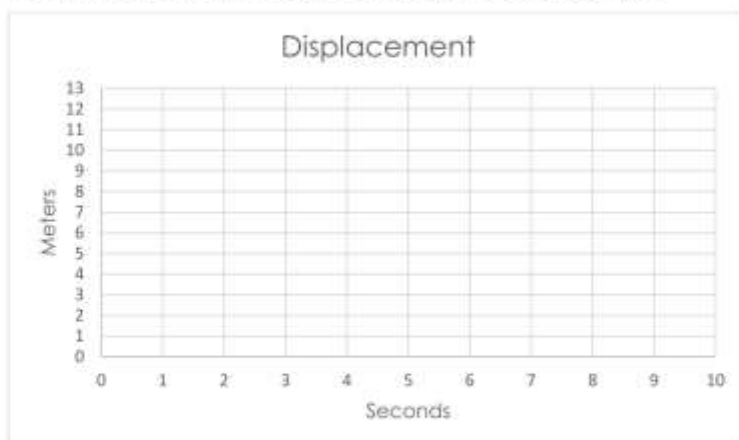
To visually understand displacement, velocity, acceleration, and their relationship, look closely at their respective graphs. Displacement has a line of



$y = x$; velocity has a line of $y = \text{constant}$; and acceleration is zero. The equation for velocity is an order lower than displacement, and acceleration is an order lower than velocity. ('Order' refers to an equation's highest exponent on a variable.) Let's say we record a different ball's motion that has an acceleration greater than zero and we obtain these following graphs:



What would be the line roughly in the displacement graph?





G. DESCRIPTION OF ACTIVITY

In this activity, you will roll a 2-liter bottle (or anything else that is big and rolls smoothly). The first step is to upload the code to the Arduino. Line up the bottle directly in front of the ultrasonic detector. Give the bottle a gentle push (that would roll the bottle no more than a meter). Once the bottle gets going, push and hold the button. Do this for 2 seconds. You now have displacement vs time data. Make sure to spot any outlying data that is clearly flawed, if there's a couple, delete them, if there's a lot, record a new set of data.

To look at your data you have recorded, go to the tool bar at the top of your window and click on "Tools", then select "Serial Monitor," and the Arduino IDE should display the data in a list in a separate window. However, to analyze and understand the data, it is best to take the data and put it into software meant for working with data; in this workshop we will use Microsoft Excel.

To do this, select all the data in the Serial Monitor and press "Ctrl" + "C" to copy it. Open excel and paste it in the cell A1. Now you might notice a problem; your data is most likely not separated into columns like the following example. There is an easy way to fix this, and we will take you step by step. 1) Go to the "Data" Tab in the top

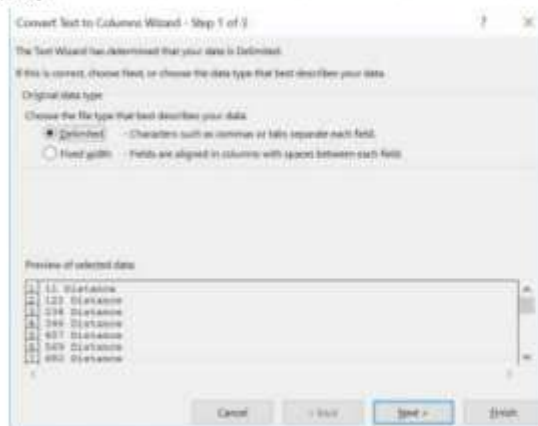
ribbon of excel. 2) Select the entire column to right of the column with combined data, right click and press "insert." This will open space for when we separate the combined column (in the example it is column B. You should have something like this: 3) Now we will separate column B. Select column B. In the Data tab, in the "Data Tools" section, click "Text to Columns." This opens up this window.

4) Make sure the "Delimited" choice is picked. Press Next.



A	B	C
Time	11 Distan	182

A	B	C	D
Time	11 Distance		182





5) Next in the Delimiters box, click "Other" and type "D," or any character that puts a separating line between data.

6) Keep pressing "Next" and "Finish" until the Wizard is completed

You can do this for any column that has too many columns of data or text that needs to be separated.

Now we are going to format our data and manipulate it so we can easily make displacement, velocity, and acceleration graphs.



1) Select row 1, right click and press "insert." This give you space to add titles to your columns of data. Title your time data column with "time(milliseconds)," and your distance data column "D(cm)."

2) Copy and Paste the entire time data column into the column that repeats the word "time" over and over again. Also delete all the text in the column that repeats the word

"distance." You should have

something like this:

A	B	C	D
Time(milliseconds)			D(cm)
11			182

3) Now we want to

convert our data to standard units. We will start out by turning our milliseconds into seconds. To do this we will make a formula in the cell to the right of the first cell with a time in it. Press "=" then click the first time in the time column, the type "/1000." The formula should read "=A2/1000." Press enter and it should return with the seconds equivalent to the box to left of it. Instead of doing this formula manually for every time, you can have Excel automatically fill in the rest of the column. In the cell you just made the formula, double click the bottom right corner when your cursor turns into a small plus sign, and it will autofill the rest of the column.

A	B
Time(milliseconds) t(s)	
3833	3.833
3935	
4037	
4139	

4) We aren't done adjusting our time data just yet. Next, we want our time data to start at 0, so our graphs will start at the origin. If you need to insert another blank column next your time columns then do that now. Title this column "t(s) (adjusted)." Just like in the last step, we will put in a formula. This formula should look like "=B2-(the number in the cell to left)." Then again, double click the bottom left corner and finish the column.

5) Next we need to convert the distance from cm to meters. In the formula bar in the adjacent column write "=D2/100," and double click the corner to fill in the full column. You should have a table similar to the example below.



	A	B	C	D	E	
1	Time(milliseconds)	t(s)	t(s)(adjusted	D(cm)	D(m)	
2		3833	3.833	0	7	0.07
3		3935	3.935	0.102	9	0.09
4		4037	4.037	0.204	10	0.1
5		4139	4.139	0.306	13	0.13
6		4240	4.24	0.407	14	0.14
7		4342	4.342	0.509	16	0.16
8		4444	4.444	0.611	18	0.18
9		4546	4.546	0.713	20	0.2
10		4648	4.648	0.815	22	0.22
11		4751	4.751	0.918	24	0.24
12		4853	4.853	1.02	26	0.26
13		4956	4.956	1.123	28	0.28

6) Now we want to have a column for velocity data. The Arduino doesn't record velocity, but we can calculate it with the data we do have. Remember velocity is the rate of change of displacement and can be described in the following equation where d_i is initial distance, d_f is the second distance, t_i is the initial time, t_f is the second time, and v is velocity: $v = \frac{d_f - d_i}{t_f - t_i}$. Follow this to make a formula in excel.

$= (E4 - E2) / (C4 - C2)$

C	D	F	F
t(s)(adjusted	D(cm)	D(m)	V
0	7	0.07	
0.102	9	0.09	$= (E4 - E2) / (C4 - C2)$
0.204	10	0.1	
0.306	13	0.13	
0.407	14	0.14	

Then again, double click the bottom right corner to have this formula go down the rest of the column. Note: You won't have data in the first cell for velocity, and you will have to delete the last, because this formula calls data from a cell above and cell below, which at least one will be blank.

D(m)	V
0.07	
0.09	0.147059
0.1	0.196078
0.13	0.197044
0.14	0.147783
0.16	0.196078
0.18	0.196078
0.2	0.196078
0.22	0.195122
0.24	0.195122
0.26	0.195122
0.28	

7) The last step is to create a column for acceleration. Recall that acceleration is the rate of change of velocity, so to create this data we will be doing the same process that we used in the last step.

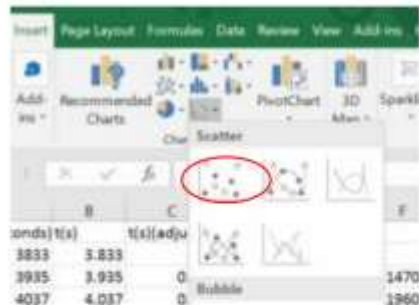


$f_a = \frac{F5-F3}{C5-C3}$				
C	D	E	F	G
t(s)(adjusted	D(cm)	D(m)	V	a
0	7	0.07		
0.102	9	0.09	0.147059	
0.204	10	0.1	0.196078	$=(F5-F3)/(I$
0.306	13	0.13	0.197044	

You have completed compiling your data, and your table should resemble the one below.

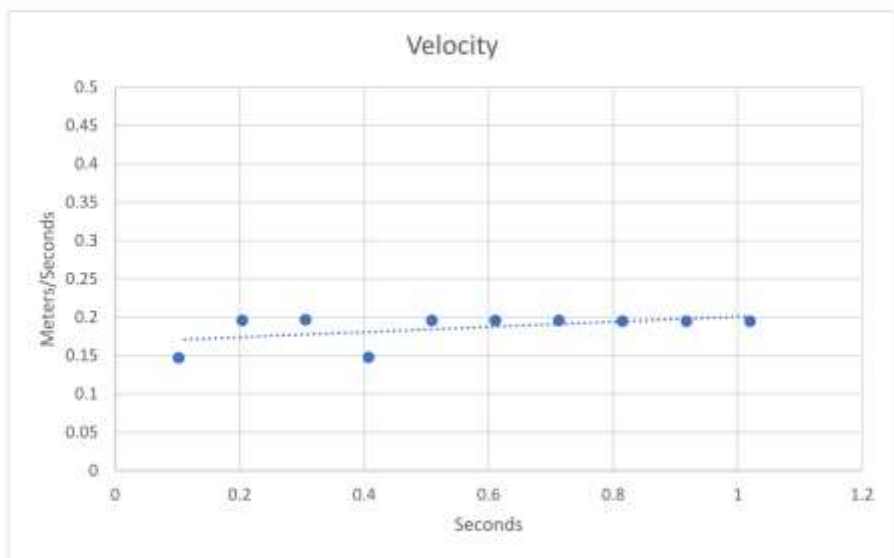
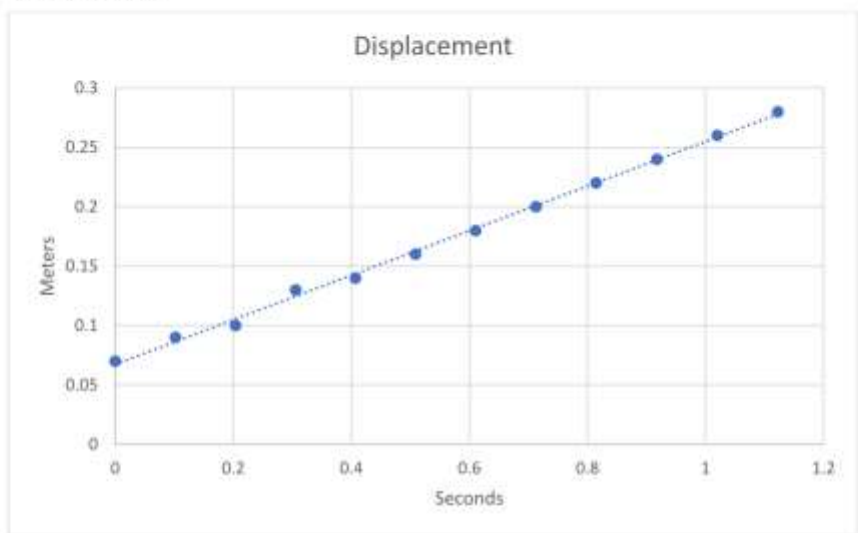
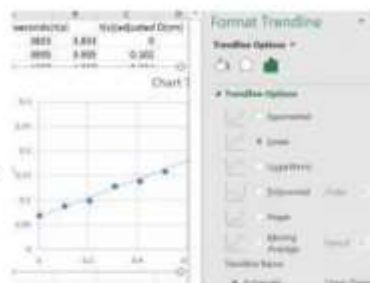
Time(milliseconds)	t(s)	t(s)(adjusted	D(m)	V	a	
3833	3.833	0	7	0.07		
3935	3.935	0.102	9	0.09	0.147059	
4037	4.037	0.204	10	0.1	0.196078	0.245027
4139	4.139	0.306	13	0.13	0.197044	-0.237907
4240	4.24	0.407	14	0.14	0.147783	-0.004758
4342	4.342	0.509	16	0.16	0.196078	0.236741
4444	4.444	0.611	18	0.18	0.196078	0.000000
4546	4.546	0.713	20	0.2	0.196078	-0.004689
4648	4.648	0.815	22	0.22	0.195122	-0.004666
4751	4.751	0.918	24	0.24	0.195122	0.000000
4853	4.853	1.02	26	0.26	0.195122	
4956	4.956	1.123	28	0.28		

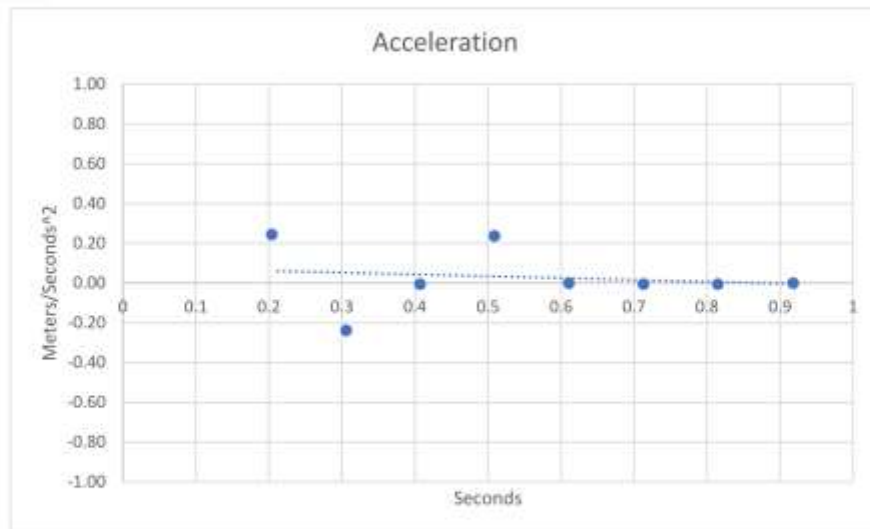
Now we are going to graph our displacement, velocity, and acceleration data. For each graph select the appropriate data, and data from t(s)(adjusted). The way to select data that's not adjacent to each other, is to select one column, then hold down the Ctrl button, then select the other column. Next, go to the "Insert" on the overhand ribbon, then in the "Charts" section, and click "Scatter Plot." Each graph should resemble a line, if it doesn't it is most likely that excel automatically has zoomed in on the data. To fix this, double click the vertical axis, and on the right-hand side you can edit the bounds, put a bigger scope, then your data will condense into a more clear line. Next, select the data points in the graph, then right click, and press "Add Trendline." Options will appear on the right, make sure "linear" is chosen





Do this process for velocity and acceleration after you have done displacement. Your displacement graph should be a straight line going up. Your velocity should be a straight line that is almost horizontal. Your acceleration graph should be a straight line hovering over the x-axis (acceleration should be close to zero). Look at the example graphs below for guidance.

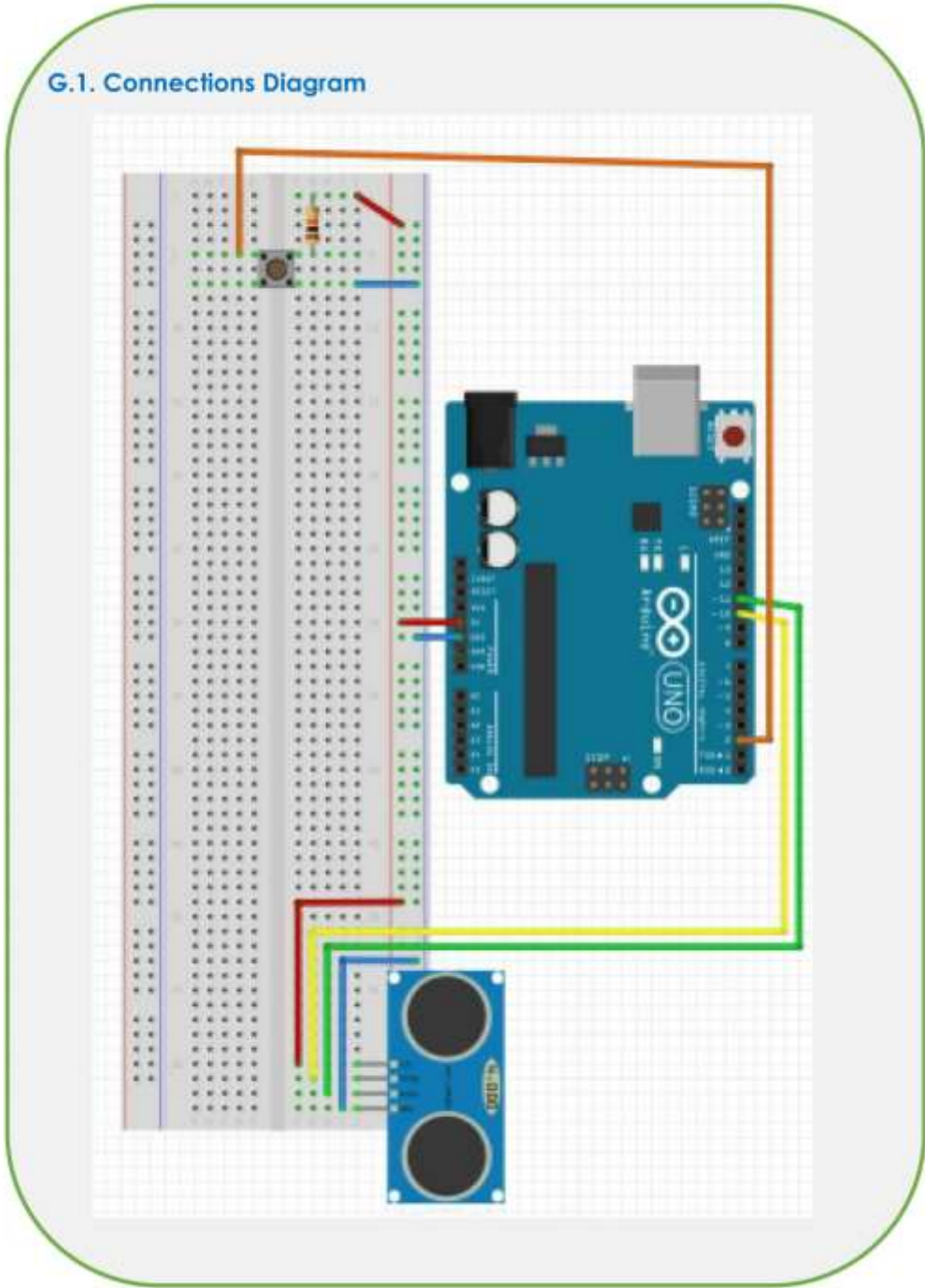




Note how close these examples experiment graphs resemble the first set of example graphs given in this workshop.



G.1. Connections Diagram





G.2. Code for Programming

```
const int trigPin = 10; // assign pin numbers
const int echoPin = 11;
const int buttonPin = 2;
int buttonState = 0;

long duration;
int distance;

unsigned long time;

void setup() {
  // Identify pins as either outputs or inputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Make it so ultrasonic detector only reads and records data when button is
  // pressed
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW){

    // Code for Ultrasonic Detector
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);

    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);

    // Distance calculation
    Distance = duration * (0.034 / 2);

    // Prints distance and time in milliseconds, for every 0.1 seconds
    Serial.print("Time: ");
    time = millis();
    Serial.print(time);
    Serial.print(" ");
    Serial.print("Distance: ");
    Serial.println(distance);
    delay(100);
  }
}
```



H. What Could Go Wrong

1) The ultrasonic detector sends beams that grow wider the farther they travel, rather than a straight precise beam. If you let your object roll too far away, the more likely you might have skewed and inaccurate data points. Having an object too small can cause the same kind of problems. Recording data only takes two seconds, don't afraid to retry several times to get good readings. If there are only a couple stand out points, you can delete them and continue on too.

2) Another issue that can cause problems is trying to do this activity with an object that doesn't roll well, this causes graphs to look very weird.

I. Challenge

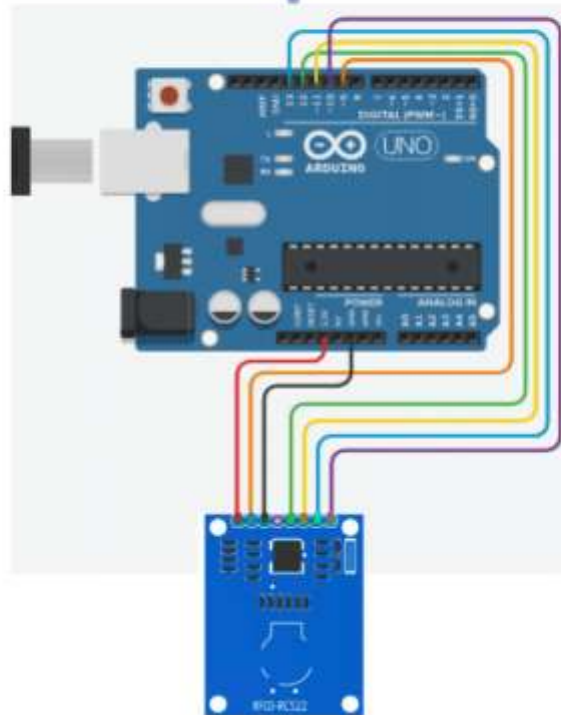
In our activity we tracked an object that had near zero acceleration. Can you do what you did today with an object with a non-zero constant acceleration? Perhaps aim the ultrasonic detector down and drop a cardboard box.

Appendix K



STORING INFORMATION USING RFID

[Arduino STEAM]





A. Title of the Project

Storing Information Using RFID

B. Subjects

Programming, Math,

C. Knowledge:

What you need to know

- How to use loops
- How to import a library
- How to use arrays

What you will learn

- How to use the RFID card reader to store information in the card
- How to use pointers
- How to pass in arrays into functions
- How to use a buffer

D. Materials Needed



Jumper cables

RFID reader and tags



E. Abstract

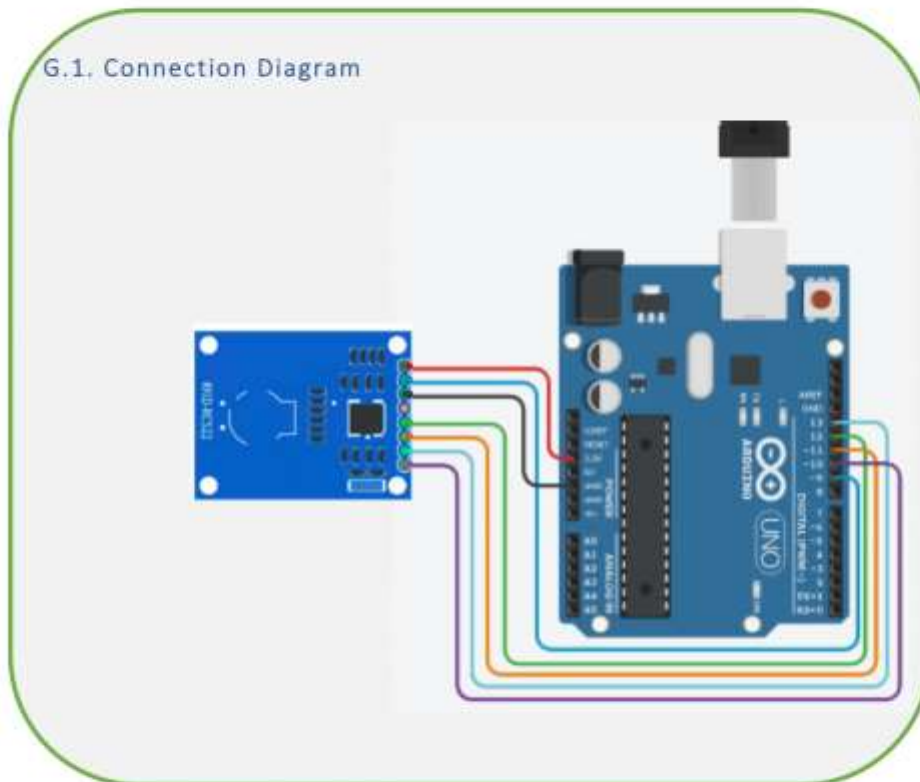
RFID (Radio-frequency identification) tags are used all around the world from door locking mechanisms, to keep track of storage in warehouses. They are, in some sense, barcodes. Instead of using an optical sensor to read them, we use radio waves. You can store a certain amount of information on RFID tags, depending on the tag. This information is stored in bytes. RFID tags alone don't emit any radio waves. It collects the waves created by a RFID reader, just like an antenna, and use these waves to transmit the information they store.

F. Context

You are told to add a "secret" message to the RFID card using the RFID reader, and print the message stored in the card on the serial monitor.

G. Activity Description

G.1. Connection Diagram



Important thing to notice in this diagram is the RQ terminal on the RFID reader is not being used. Be sure that all of the wires you are using are working. You should get a red light on the reader if you connect the power correctly. After connecting the RFID reader, go to the following link, and download the .zip library.

<https://github.com/miguelbalboa/rfid>

Import the library by going to the taskbar "Sketch -> Include Library -> Add .Zip Library..." after importing the library go: "File -> Examples -> MFRC522 -> ReadAndWrite" Read the code and try to figure what steps it takes to read the data on the RFID tag, write something new on it, and read the newly stored data. Don't worry if there are many lines of code you don't understand. Just figure out what the blocks of code do.



After doing that copy the activity code and compare it to the “ReadAndWrite” program you just read. What differences do you see?

G.2. RFID Reader and RFID Tag

RFID is a technology that started commercial use in the 1970’s as an identification method for cars going through tolls. An RFID reader is a device used to read the information stored in a RFID tag. An RFID tag consists of an integrated circuit that modulates and demodulates the radio frequencies, and an antenna to transfer the information to the RFID reader. A passive RFID tag can only work if there is a RFID reader in its range, because it cannot create radio waves by itself. However active RFID tags have an on-board battery, which allows it to periodically transmit the data stored in it. The data in RFID’s are stored in “blocks” and “sectors”. This means that the data is divided into smaller chunks with certain keys stored at the start of the chunk to identify it. This makes it easier to find where the information is stored instead of looking through all of the data in it. The tags we will be using have a storage capacity of 1024 bytes.

G.3. Program Code

```
#include <SPI.h>

#include <MFRC522.h>

constexpr uint8_t RST_PIN = 9;    // Configurable, see typical pin layout above
constexpr uint8_t SS_PIN = 10;   // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
MFRC522::MIFARE_Key key;

void setup() {
  Serial.begin(9600); // Initialize serial communications with the PC
  while (!Serial);   // Do nothing if no serial port is opened (added for Arduinos
  based on ATMEGA32U4)
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card

  // Prepare the key (used both as key A and as key B)
  // using FFFFFFFFh which is the default at chip delivery from the factory
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}
```




```
Serial.println(F("Scan a MIFARE Classic PICC to demonstrate read and write."));
Serial.print(F("Using key (for A and B):"));
dump_byte_array(key.keyByte, MFRC522::MF_KEY_SIZE);
Serial.println();

Serial.println(F("BEWARE: Data will be written to the PICC, in sector #1"));
}

/**
 * Main loop.
 */
void loop() {
    // Look for new cards
    if ( ! mfrc522.PICC_IsNewCardPresent() )
        return;

    // Select one of the cards
    if ( ! mfrc522.PICC_ReadCardSerial() )
        return;

    // Show some details of the PICC (that is: the tag/card)
    Serial.print(F("Card UID:"));
    dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
    Serial.println();
    Serial.print(F("PICC type: "));
    MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
    Serial.println(mfrc522.PICC_GetTypeName(piccType));

    // Check for compatibility
    if (   piccType != MFRC522::PICC_TYPE_MIFARE_MINI
        && piccType != MFRC522::PICC_TYPE_MIFARE_1K
        && piccType != MFRC522::PICC_TYPE_MIFARE_4K ) {
        Serial.println(F("This sample only works with MIFARE Classic cards."));
        return;
    }
}
```



```
// In this sample we use the second sector,  
// that is: sector #1, covering block #4 up to and including block #7  
byte sector      = 3;  
byte blockAddr  = 14;  
byte dataBlock[16] = {};  
  
// the spaces at the end of the message is to have 45 characters in the message  
String message = "We are storing this message in the card    ";  
String subs[3]; // divided message into 15 character strings  
divideString(message, subs, 3);  
byte trailerBlock = 15;  
MFRC522::StatusCode status;  
byte buffer[18];  
byte size = sizeof(buffer);  
  
// Authenticate using key A  
status = (MFRC522::StatusCode)  
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key,  
&(mfrc522.uid));  
  
if (status != MFRC522::STATUS_OK) {  
    Serial.print(F("PCD_Authenticate() failed: "));  
    Serial.println(mfrc522.GetStatusCodeName(status));  
    return;  
}  
  
// Show the whole sector as it currently is  
Serial.println(F("Current data in sector:"));  
mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);  
Serial.println();  
  
// Authenticate using key B  
status = (MFRC522::StatusCode)  
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B, trailerBlock, &key,  
&(mfrc522.uid));  
  
if (status != MFRC522::STATUS_OK) {  
    Serial.print(F("PCD_Authenticate() failed: "));  
    Serial.println(mfrc522.GetStatusCodeName(status));  
    return;  
}  
}
```



```
// Write data to the block
Serial.print(F("Writing data into block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
Serial.println("");
Serial.print("Data: ");
Serial.print(message);
Serial.println("");
for(int i = 0; i < 3; i++){
    subs[i].toCharArray(dataBlock, 16);
    status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(blockAddr, dataBlock, 16);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("MIFARE_Write() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
    }
    blockAddr--;
}
blockAddr = 14;

// Read data from the block (again, should now be what we have written)
for(int j = 0; j < 3; j++){
    subs[j].toCharArray(dataBlock, 16);
    status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer, &size);
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("MIFARE_Read() failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
    }
    Serial.println("");
    Serial.print(F("Data in block ")); Serial.print(blockAddr);
    Serial.println(F(":"));
    Serial.write(buffer, 16); Serial.println();

    // Check that data in block is what we have written
    // by counting the number of bytes that are equal
    byte count = 0;
    for (byte i = 0; i < 16; i++) {
        // Compare buffer (= what we've read) with dataBlock (= what we've written)
```



```

    if (buffer[i] == dataBlock[i])
        count++;
    }
    Serial.print(F("Number of bytes that match = ")); Serial.println(count);
    if (count == 16) {
    } else {
        Serial.println(F("Failure, no match :-("));
        Serial.println(F(" perhaps the write didn't work properly..."));
    }
    blockAddr--;
}
blockAddr = 14;

// Dump the sector data
Serial.println(F("Current data in sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);
Serial.println();

// Halt PICC
mfrc522.PICC_HaltA();
// Stop encryption on PCD
mfrc522.PCD_StopCrypto1();
}

/**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void dump_byte_array(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}

void divideString(String string, String *temp, int tempLength){
    String sub;

```

```

for(int i = 0; i <= templength; i++){
    if(i == templength){
        sub = string.substr(i*15);
        temp[i] = sub;
    }
    sub = string.substr(i*15, ((i+1)*15));
    temp[i] = sub;
}
}

```

G.4. Pointers

Pointers are a programming concept used to use memory effectively and efficiently. A pointer is basically a variable that "points" to a value that is stored in memory. For example:

```


int p;
int value;
int num = 4;
p = &num;
value = *p;

```

In this block of code, we have a pointer called "p" which points to the address in memory where the variable "num" is stored, and the variable "value" has the same value that p points at, which is 4. So here are few important details in the code. You can tell that p is a pointer because it has a "*" before it. This tells the computer that it is a pointer and not a simple integer variable. Then we assign &num to the pointer because putting an "&" sign before a variable gives us the address of where that variable is stored in memory. If we didn't put the "&", the pointer would point to the variable, which would give us a value like "2531", which would be the address of the variable instead of the value stored in that address. We also use the "*" again when we are assigning a value to the variable "value", because we want the value the pointer is pointing to. If we just used "p", value would be equal to the address of the variable num.

G.5. Passing In an Array To a Function

Passing an array is a bit more complicated than passing a variable. Simply because when you call an array, you just have a pointer that points to the first element of the array. That is why you have to specify which element you want in the array when getting values. When you want to write a function that modifies an array, which was initialized outside of that function, you have to pass in the pointer to the first element, and the size of the array. This is to not access values stored in memory, which is not in the array. Here is an example from the activity code:



```

void divideString(String string, String *temp, int tempLength){
    String sub;
    for(int i = 0; i <= tempLength; i++){
        if(i == tempLength){
            sub = string.substring(i*15);
            temp[i] = sub;
        }
        sub = string.substring(i*15, ((i+1)*15));
        temp[i] = sub;
    }
}

```

This is a function which takes in a string and an array, divides the string into 15 character strings, and store it in the array that was passed in. The pointer "***temp**" is a pointer to the first element of the array. This allows us to change the value stored in an array outside the function by changing the value stored in a specific address in memory. We use "**tempLength**" as a limit to our for() loop to not change any values outside of the array, because another variable might be stored in the next address in memory. This is called a memory leak.

G.6. Buffers

A buffer is a word we use to describe a data structure, usually an array, which temporarily stores information in the memory. It is usually used when transferring data from one place to another. For example we can store all the data we retrieve from an input device in a buffer before we put it in a database. They are also useful when transferring data in between processes in a computer. Buffers are useful because they act as a "middle-step" in a data transfer process. Here is an example of buffers from the activity code:

```

void dump_byte_array(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}

```

This is a function which is called to print the information that was read from the card, onto the serial monitor. When reading the data on the card, we store it in a buffer so that we can print it or compare it to our data block to see if they match. After finishing any operation we want to make on the buffer, we empty the buffer and get fill it with a new block of data from the card. This helps us because we don't have to read all the data on the card, store it in a huge array and print them all at once. We can just read and print block by block, and only have to use one small array.



H. What Could Go Wrong

1. Connecting the RFID reader is difficult. You have to make sure all the cables are touching the metal holes on the RFID reader. You may want to solder it to the breadboard if you cannot get it to connect properly. You can test if it is connected properly by executing the "DumpInfo" example
2. Check to see if your RFID reader is working by only connecting the 3.3V pin and the ground pin. A red LED should light up on the reader. If not it means that it can't get power. This could be caused by the cable, the 3.3V pin, or the reader itself. You can easily change the cable if the problem is caused by the cable. If the 3.3V pin is causing the problem you can use the 5V pin and connect a 10KΩ resistor before connecting it to the RFID reader. If the problem is caused by the reader, you have to change the reader.
3. If you are not able to write any info into the card. Check the error message you get. If you get a key authentication error, you may have overwritten the key in the card which would cause that section in the card to be unusable. If you are getting connection timeout error, it might be because of a cable disconnection, or the reader was not able to read the card correctly. Make sure that you are keeping the card stable and very close to the reader.

I. Challenge

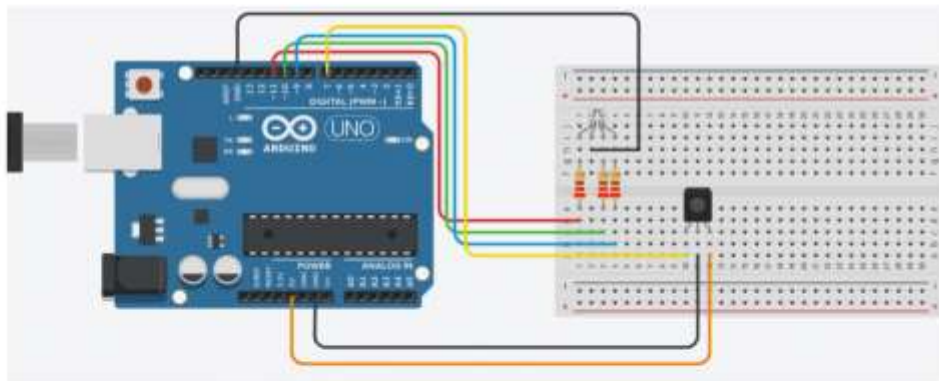
After carefully reading the code, change it to store a different message in a different section of the card. Be careful because if you change the last 6 bytes stored in the first block of a section, you will lose the key to authenticate that section, which means you will not be able to read or write on that section. You can also try to store a string longer than 45 characters. You will need a way to divide the string and store in in different sections.

Appendix L



CONTROLLING A RGB LED WITH A REMOTE

[Arduino STEAM]





A. Title of the Project:

Controlling a RGB LED With a Remote

B. Subjects

Programming, Math, Electrical Engineering

C. Knowledge

What you need to know

- How to use loops
- How to create new functions
- How to use switch cases

What you will learn

- How to import libraries
- How to use a RGB LED
- How to connect a remote to control the Arduino
- How to create new data types (structures)

D. Materials Needed



Jumper cables



Remote



RGB LED



IR receiver



E. Abstract

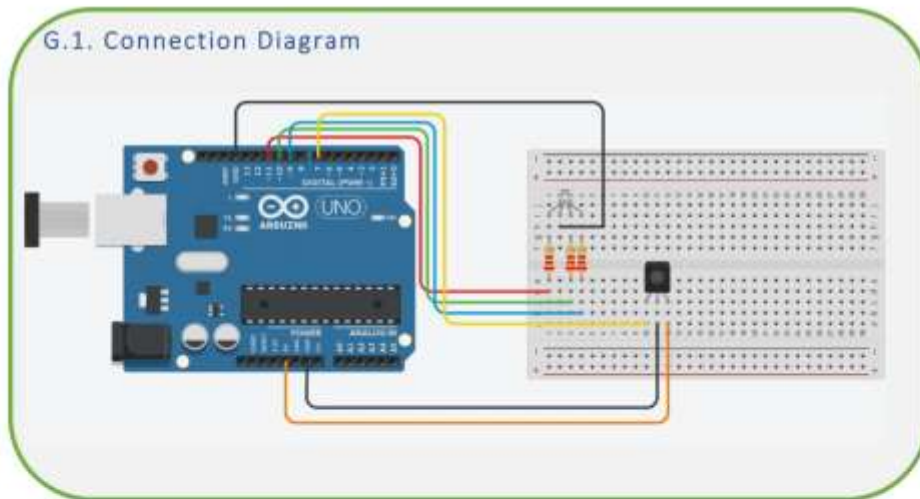
RGB LED's are used in many situations from decorations to TV and other displays. The idea behind RGB LED's is to create 256^3 different colors by only mixing red, green, and blue. The total number of colors you can create is 256^3 because the LED has values between 0 and 255 for the main colors red, green, and blue. This value determines how bright/powerful that color will be, where 0 means off and 255 means maximum brightness. You will also use a remote controller which uses infrared (IR) waves to send information about which button is pressed. IR waves are a type of electromagnetic wave, which has a wavelength between 700 nanometers and 1 millimeter. This makes the waves invisible to the human eye because it is out of the visible light spectrum.

F. Context

Your friend wants to throw a party and asked you to help with decorations. He wants to have a remote controlled light, which can create sequence of colors to make people dance and have fun.

G. Activity Description

G.1. Connection Diagram



Important things to notice in this diagram are the red, green, and blue wires corresponding to red, green, and blue terminals of the LED. For the IR sensor, when you are facing the sensor side, the left terminal transmits the signal, middle one is ground, and right one should be connected to 5V.



G.2. RGB LED

This part is called RGB LED because it has three small light emitting diodes of three different colors; red, green, and blue. We can use these three colors to create most of the colors we know. We can create exactly 16777216 different colors because we have three colors that has a brightness value between 0 and 255, where 0 means it is not emitting any light and 255 means it is emitting the maximum amount of light. This creates 256^3 different color combinations. The current passing through the wire controls the intensity of the colors. If the current is higher for one color, that color will be brighter. There are two types RGB LED's. One is just an LED with four terminals sticking out of it. The other one is the same LED attached to a breakout board. The way you connect them are also different. The LED only type has one terminal, which is longer than all the other terminals. That one should be connected to the ground. The rest of the terminals are for red green and blue. You can see which one are the corresponding terminals on the diagram below. The one attached to a breakout board is easier to connect because it has the letters corresponding to R, G, B, and "-" for ground.

G.3. Remote and IR Sensor

Infrared radiation is everywhere. Light bulbs, the sun, humans, basically almost everything that produces heat produces infrared radiation. The IR sensor easily picks up most of the IR radiation around it. This is called environmental noise because it is not the signal we want to pick up. Therefore, the sensor uses a band-pass filter, which greatly decreases the value of IR radiation it reads that is out of the "band" range. So in other words we want the sensor to only read the same frequency range as the IR radiation emitted by the remote. The remote can be seen as a board with bunch of buttons and an IR transmitter. The IR radiation emitted by the remote includes information about which button is pressed, and keeps sending information if the button is held down. This information is converted into something called a hex code. Every button has a different hex code, so we have to convert the hex codes into the symbols on the remote to understand which button is pressed.

G.4. Program Code

```
#include <IRremote.h>

const int RECV_PIN = 7;
IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long key_value = 0;

int redPin = 5;
int greenPin = 10;
int bluePin = 9;
```



```
struct Color{
  int red;
  int green;
  int blue;
};

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(9600);

  // IR reciever
  irrecv.enableIRIn();
  irrecv.blink13(true);
}

void loop() {
  if (irrecv.decode(&results)){
    if (results.value == 0xFFFFFFFF)
      results.value = key_value;

    switch(results.value){
      case 0xFFA25D:
        Serial.println("CH-");
        break;
      case 0xFF629D:
        Serial.println("CH");
        break;
      case 0xFFE21D:
        Serial.println("CH*");
        break;
      case 0xFF220D:
        Serial.println("|<<");
        break;
      case 0xFF02FD:
```



```
Serial.println(">>|");
break ;
case 0xFFC23D:
Serial.println(">|");
break ;
case 0xFFE01F:
Serial.println("-");
break ;
case 0xFFA857:
Serial.println("+");
break ;
case 0xFF906F:
Serial.println("EQ");
break ;
case 0xFF6897:
Serial.println("0");
break ;
case 0xFF9867:
Serial.println("100+");
break ;
case 0xFFB04F:
Serial.println("200+");
break ;
case 0xFF30CF:
spectrum();
break ;
case 0xFF18E7:
strobe();
break ;
case 0xFF7A85:
fade(randColor());
break ;
case 0xFF10EF:
oneByOne(randColor());
break ;
case 0xFF38C7:
```



```
    blinkColor();
    break ;
    case 0xFF5AA5:
    Serial.println("6");
    break ;
    case 0xFF428D:
    Serial.println("7");
    break ;
    case 0xFF4A85:
    Serial.println("8");
    break ;
    case 0xFF52AD:
    Serial.println("9");
    break ;
  }
  key_value = results.value;
  irrecv.resume();
}
}

Color randColor(){
  randomSeed(millis());
  Color randRGB = {
    random(0, 255),
    random(0, 255),
    random(0, 255),
  };
  return randRGB;
}

void showColor(Color color){
  analogWrite(redPin, color.red);
  analogWrite(greenPin, color.green);
  analogWrite(bluePin, color.blue);
}
```



```
void spectrum(){
  Color spectrum = {0,0,0};
  while(spectrum.red < 255){
    spectrum.red += 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.green < 255){
    spectrum.green += 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.green == 255 && spectrum.red != 0){
    spectrum.red -= 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.blue < 255){
    spectrum.blue += 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.blue == 255 && spectrum.green != 0){
    spectrum.green -= 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.red < 255){
    spectrum.red += 5;
    showColor(spectrum);
    delay(40);
  }
  while(spectrum.green < 255){
    spectrum.green += 5;
    showColor(spectrum);
    delay(40);
  }
}
```



```
}  
while(spectrum.green!=0 && spectrum.blue!=0 && spectrum.red!=0){  
    spectrum.red -= 5;  
    spectrum.green -= 5;  
    spectrum.blue -= 5;  
    showColor(spectrum);  
    delay(40);  
}  
}  
  
void strobe(){  
    digitalWrite(redPin, HIGH);  
    digitalWrite(greenPin, HIGH);  
    digitalWrite(bluePin, HIGH);  
  
    delay(10);  
  
    digitalWrite(redPin, LOW);  
    digitalWrite(greenPin, LOW);  
    digitalWrite(bluePin, LOW);  
  
    delay(20);  
  
    digitalWrite(redPin, HIGH);  
  
    delay(20);  
  
    digitalWrite(redPin, LOW);  
    digitalWrite(greenPin, HIGH);  
  
    delay(20);  
  
    digitalWrite(greenPin, LOW);  
    digitalWrite(bluePin, HIGH);  
  
    delay(20);
```




```
digitalWrite(bluePin, LOW);
}

void fade(Color color){
  for(int i = 0; i <= 255; i+=5){
    if(i < color.red){
      analogWrite(redPin, i);
    }
    if(i < color.green){
      analogWrite(greenPin, i);
    }
    if(i < color.blue){
      analogWrite(bluePin, i);
    }
    if(i>color.red && i>color.green && i>color.blue){
      break;
    }
    delay(40);
  }
}

void oneByOne(Color color){
  analogWrite(redPin, LOW);
  analogWrite(greenPin, LOW);
  analogWrite(bluePin, LOW);
  delay(50);

  analogWrite(redPin, color.red);
  delay(300);
  analogWrite(greenPin, color.green);
  delay(300);
  analogWrite(bluePin, color.blue);
  delay(300);
}
```

```

analogWrite(redPin, LOW);
analogWrite(greenPin, LOW);
analogWrite(bluePin, LOW);
delay(300);

showColor(color);
}

void blinkColor(){
  showColor(randColor());
  delay(500);
  analogWrite(redPin, LOW);
  analogWrite(greenPin, LOW);
  analogWrite(bluePin, LOW);
  delay(100);
}

```

G.5. Importing a Library

All programming languages and development environments have programs and tools written by someone else, which you can use in your own program. This cluster of program files is called a library. Libraries can be extremely helpful if you don't know how to use a specific part or program. The files in the library include helpful functions, data types, and other useful code. Here is how we import a library to use for the IR sensor. First of all go to this site and download the .zip file:

<http://z3t0.github.io/Arduino-IRremote/>



Arduino IRremote

Infrared remote library for Arduino: send and receive infrared signals with multiple protocols

tar.gz .zip

IRremote Arduino Library

build [view](#)

github [join chat](#)

This library enables you to send and receive using infra-red signals on an Arduino.

Check [here](#) for tutorials and more information.

Click to download



After downloading the .zip file go into the Arduino IDE and go;

“Sketch -> Include Library -> Add .Zip Library...”

Then find the .zip file you downloaded and double click on it. This makes it possible for you to use the functions and files in the library. After doing that we need to call the file we want to use at the start of the program. In the activity we are trying to use a file called “IRremote.h”, so at the top of the program we have the line:

```
#include <IRremote.h>
```

This enables us to use the functions and data types in “IRremote.h” file.

G.6. Structures/Data Types

Data types are one of the most important aspects of programming. Every variable has a data type. This could be integer, string, array, float... These are examples of basic data structures that you have been using. Data structures are important because they all keep a certain data in different ways. For example an integer (int) variable can only be a number between -2147483648 and 2147483648, because it has to be represented by 4-bytes (where the first bit determines if it is negative or not). However, if you are not sure what data type to use, or think you want to keep track of one piece of data that includes different data types, you can use structures. Structures are basically data structures you can create. For example if I was collecting information from people about their name, height in meters, age, and job, I can create a structure as:

```
struct Person{
    String name;
    float height;
    int age;
    String job;
};
```

It is important to create the structure before creating a variable that uses that structure as a data type. This structure allows us to create a variable with the information we mentioned.

Here is an example of such a variable:

```
Person george = {"George", 1.75, 17, "student"};
```

Now we have a variable “george” with name, height, age, and job. But what can we do with this information. We can’t do much with just the variable, but we can access the information in the variable as:

```
String name = george.name;
int age = george.age;
int height = george.height;
String job = george.job;
```



Basically we can access the information stored in the variable "george" by putting a period after the variable call and specify which data we want to access in it. The **name of the data** depends on what you named it while creating the structure.

In the activity we have a structure called "Color" which has a red, green, and blue value, because when we create a color and want to show it using the RGB LED we have to give the red, green, and blue values. The structure is defined as:

```
struct Color{
    int red;
    int green;
    int blue;
};
```

So when we want to create a color we can just use this structure and give it three values between 0 and 255. Such as:

```
Color red = {255, 0, 0};
Color purple = {155, 0, 200};
```

G.7. Remote – IR Sensor Code

After importing the library we are going to use some functions from it to understand what the sensor is reading. We have three pieces of code to initialize needed variables, activate the sensor, and find out which button is being pressed. Here is the initialization piece:

```
const int RECV_PIN = 7;
IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long key_value = 0;
```

Here, we use a data type which was defined in IRremote.h. It is called "IRrecv" and we give the pin number as a parameter to assign that pin to the receiver. It serves the same purpose as the pinMode() function, but for an IR receiver. Decode_results is also a new data type which will take the IR value readings, decode it, and store it in a variable called "results". The next piece of code, which is activating the sensor:

```
irrecv.enableIRIn();
irrecv.blink13(true);
```

This part is inside of the setup() function, and it enables the IR receiver. The last piece is to understand which button is pressed:

```
if (irrecv.decode(&results)){
    if (results.value == 0xFFFFFFFF)
        results.value = key_value;
```



```
switch(results.value){  
  case 0xFFA25D:  
    Serial.println("CH-");  
    break;  
  case 0xFF629D:  
    Serial.println("CH");  
    break;  
}
```

This piece is inside the `loop()` function and has many switch cases, so this is just a part of it. The first if statement checks if the receiver is collecting results. The second if statement checks a button is being held down. Finally the switch statement checks which button is pressed. Each case is the hex code for a specific button on the remote, and prints the corresponding symbol on the remote. In the activity code, buttons from 1 to 5 are mapped to run a certain function which outputs a sequence of colors.

H. What Could Go Wrong

1. Be sure to copy the code exactly. It is considerably long, and if you miss a single curly bracket, it will not run properly.
2. Be sure that you have a single resistor for each color terminal on the RGB LED. If you connect a power source directly to the LED you might break it because it will have infinite current going through it.
3. Make sure you downloaded the library correctly and included the .zip file in the libraries using the Arduino IDE.

I. Challenge

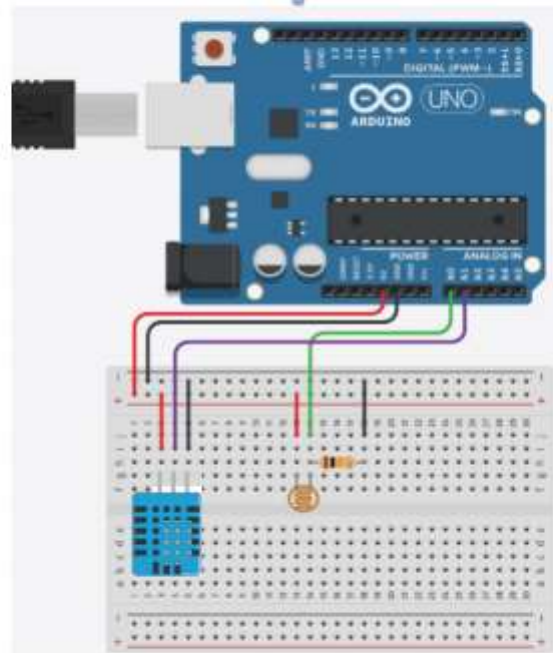
Read the code for all of the functions for different light sequences, change them and create new ones. You can use the functions like `randColor()` to create random colors, or use functions inside of functions. Be creative with it. You can also add more parts like more LED's or a buzzer to have two or more outputs by just pressing a single button.

Appendix M



MAKING A CLIMATE CONTROL SENSOR

[Arduino STEAM]





A. Title of the Project

Making a Climate Control Sensor

B. Subjects

Programming, Data Science

C. Knowledge

What you need to know

- How to import a library
- How to read analog input
- How to print data collected on the Serial Monitor

What you will learn

- How to read from a serial input, and write it to a file using Python
- How to permanently store data collected from an Arduino

D. Materials Needed



Jumper cables



DHT-11



Photoresistor



10kΩ resistor

E. Abstract

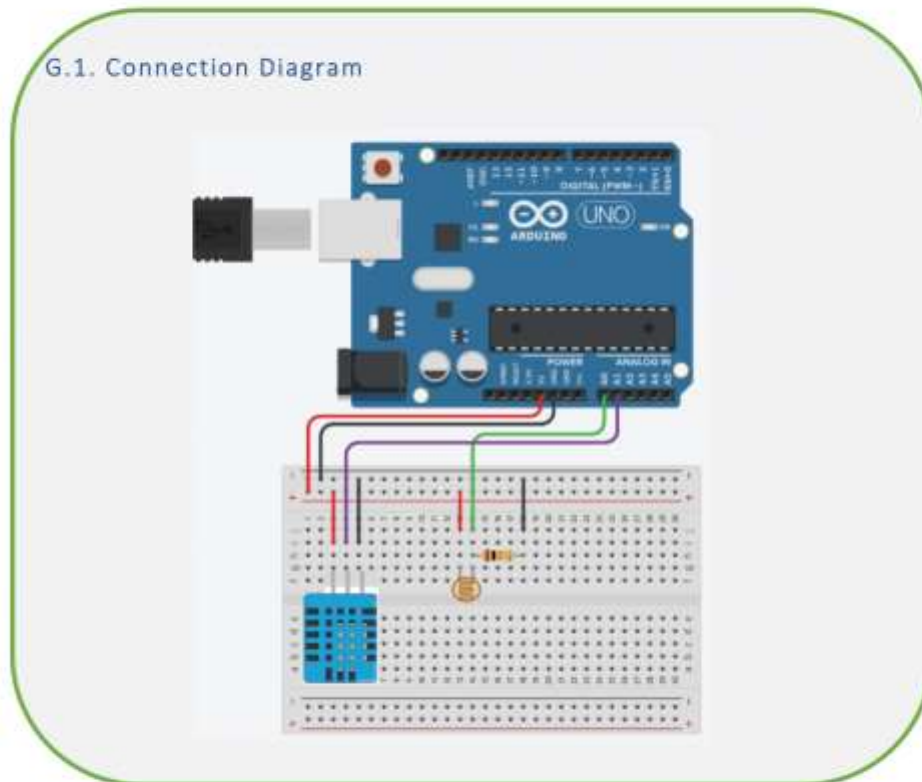
Data collection is a very important aspect of science. It helps us prove/disprove our hypothesis. With advancements in sensor technologies we are able to collect extremely accurate data in lightning fast speeds. The way we store all this data is as important as the data itself. If we had a page with bunch of numbers of it, would be hard to make any sense out of those numbers. That is why we have to explain what the data we are collecting means, and store it in an organized fashion.

F. Context

You are asked to make a climate control machine for a greenhouse, which collects information about humidity, light intensity, temperature, and date. You should collect all this data and store it in a .txt file in your computer.

G. Activity Description

G.1. Connection Diagram



After connecting all the parts according to the diagram above, copy and paste the URL below in your internet browser to download the library for DHT-11.

https://www.brainy-bits.com/wp-content/uploads/2017/12/DHT_Library.zip

Import the library into the Arduino IDE by going to the taskbar "Sketch -> Include Library -> Add .Zip Library..."

G.2. DHT-11

DHT-11 is a sensor that collects information about temperature and humidity. Inside the sensor there is a capacitive humidity sensor that measures the humidity of the air surrounding it, and a thermistor, which measures the temperature. It is a very simple and inexpensive part that can collect fairly accurate data. If you want a more accurate sensor,



which has a wider range, try using DHT-22. It is basically a better but bigger and more expensive sensor compared to the DHT-11. The DHT library helps us because instead of reading an analog input and converting it to Celsius, we can just call `DHT.temperature` and it will return the temperature in Celsius.

G.3. Photoresistor (LDR)

A photoresistor, also called a Light Dependant Resistor, is a type of resistor that doesn't have a fixed resistance value. The resistance depends on the amount of light it is exposed to. The resistance is directly proportional with the amount of light. This means that the resistance increases as the amount of light increases. We have to use a resistor with a fixed resistance with the photoresistor to not have a short circuit when there is very small amount of light exposed to the photoresistor. We can use this part as a sensor by connecting a cable between the photoresistor and the normal resistor. This reading will give us an idea about the potential difference/voltage of the circuit at that location. However, this will be a reading between 0 and 1024 because it is an analog reading. This means that we have to convert that value into Volts.

G.4. Part 1: Arduino

Program Code Part 1

```
#include "dht.h"

#define dht_apin A1

dht DHT;

int photoPin = A0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  float photo = analogRead(photoPin);
  float vout = (photo/1024)*5;
  float lux = ((2500/vout)-500)/10;
  DHT.read11(dht_apin);
```



```
Serial.println("#####");
Serial.print("Light index: ");
Serial.print(lux);
Serial.println(" Lux");
Serial.println("");

Serial.print("Temperature: ");
float celcius = DHT.temperature;
Serial.print(celcius);
Serial.print("°C");
float fahrenheit = (celcius * (1.8)) + 32;
Serial.print(" / ");
Serial.print(fahrenheit);
Serial.println("°F");
Serial.println("");

Serial.print("Humidity: ");
Serial.print(DHT.humidity);
Serial.println("%");
Serial.println("");

delay(10000);
}
```

G.5. Units

Units are one of the core concepts in any type of science subject, because they make it possible for us to compare data we collected to data that someone else collected. There are many standardized units of measurements for many different types of data. For example we use Meters or Feet for measuring length, Celsius or Fahrenheit for temperature, Grams or Ounces for measuring weight... If we didn't have any units at the end of the data we collect, we have no way to know what that number means. In this activity we are using Lux to measure the amount of light, Celsius and Fahrenheit for temperature, and percentage to measure the humidity in the air.

G.6. Part 2: Python

In this part of the activity you are going to download and use python 3.7 to read the Serial port of your computer. Then write what we read to a text file and store it in our hard drive, so that we have a permanent file with all the data we have collected using the Arduino.

First of all we have to download Python on our computer. So use the following link to go to Python's website and click "Download Python 3.7" or above.

<https://www.python.org/downloads/>



After downloading the file, click on it to run. Click "Continue" a few times and click agree when you are asked to agree to the terms and conditions. Click "Continue" one more time to start the installation. It is going to ask for your password to install so you have to enter your password for the user you are using. When you get a message indicating that the installation is done, click "Finish" and you are done downloading Python. But you are not done yet. The rest of the activity will be different depending on what operating system you are using. Mac or Windows.

G.6.1. For Mac

Open up "Terminal". If it is not on your Dock, you have to open up Finder and go "Applications -> Utilities -> Terminal". Double click on it to run it, and you should get a small window like this. The text on it will be different since it will be showing information from your computer,



```
Last login: Thu Oct 4 13:49:26 on ttys000
Cems-mac-air-4:- cemalendar$
```

In the terminal write the following line and press enter:

```
python3 -V
```

This is to check if you downloaded Python correctly. If it says Python 3.7 or above it means you have it downloaded correctly. If not, go back and follow the installation process again.

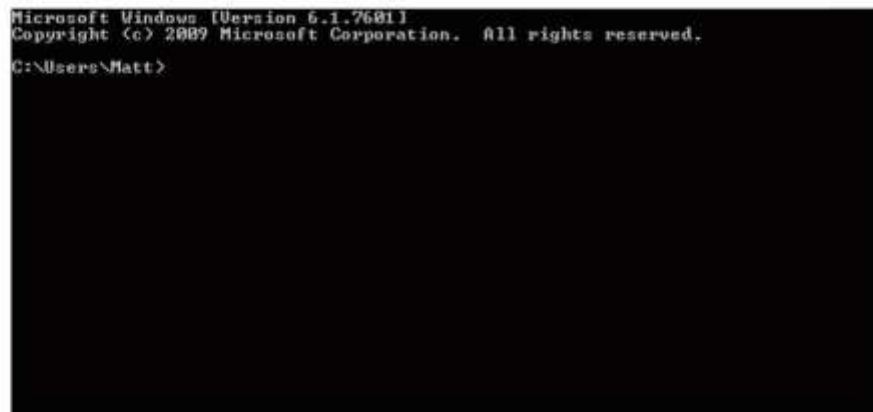
Afterward you have to install Pyserial, which is a library we need for reading the serial input from the Arduino, so use the following command in the Terminal:

```
pip3 install pyserial
```

This should give you a success message. If not try closing the terminal and re-opening it.

G.6.2. For Windows

After installing Python 3.7 or above you should restart your computer. When you turn it back on, go ahead and open up the Command Prompt. You can do this by searching "cmd" in the Start menu. You should get a screen as such when you run Command Prompt:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Matt>
```

Now go ahead and type in the following line and press enter:

```
python3 -V
```

This allows you to see the version of python downloaded on your computer. If you got a message saying "Python 3.7" or above after pressing enter it means that you have downloaded



Python correctly. If you got a version lower than 3.7 or got an error message it means that you didn't download Python correctly and you should go back to the installation process and try it again. After making sure you have Python installed correctly, go ahead and type in the following line to the Command Prompt and press enter. This installs a library called Pyserial, which we need to read the serial input coming from the Arduino.

```
pip3 install pyserial
```

This should give you a success message. If not try closing the Command Prompt and re-opening it.

G.7. Continuing Part 2 for Both OS's

Now go to the following link and download the file called "ReadSerial.py" and place the file in your Desktop.

<https://github.com/calemdar/ReadSerial>

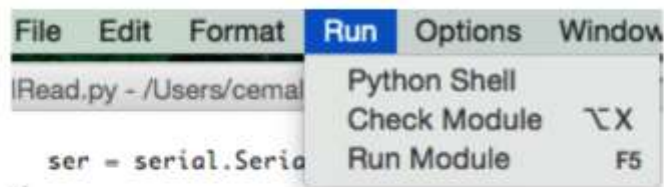
It is important that you place it in the Desktop because the program will create a file called output.txt in the same folder you place the Python file. This output.txt file will be the file we store the information we read from the Serial input coming from the Arduino. Now left-click on the "ReadSerial.py" file you just opened and click "Open with -> IDLE". You should be able to read the code in the file. If you know which port your Arduino is connected to, go ahead and write it in the quotation marks for the variable PORT shown below.

```
#####
PORT = '' # write port here, i.e. 'COM3'
#####
```

If you don't know the port your Arduino is connected to, you can find it out by going into the Arduino IDE and check the bottom of the Arduino window. It should say "Arduino/Genuino UNO on" then it shows the port you are using.



If you cannot find it, the program will try to find it itself when you run it. Now make sure that your Arduino is connected and working properly, and close the Arduino IDE. To run the Python program go to the IDLE window, save the file, then on the taskbar click on "Run -> Run Module".



If you get the message "Starting to listen to the serial port", it means that the program is working correctly. There should be a new file called output.txt with the readings. There might not be too many readings when you open the file since we only collect data every 10 seconds. Leave the program running for a minute or so and reopen output.txt to see more readings. If you got a message "No serial connection found!" it means that the Arduino is not connected or you put in the port wrong. Check which port the Arduino is using again, or just leave it blank and let the program try to find it.

H. What Could Go Wrong

1. Make sure your Arduino is collecting data correctly by checking the Serial Monitor. Check if your sensors are working correctly by changing the light and temperature around it.
2. If the Python program is not working properly, it could be due to a connection error with the serial port. Make sure you put in the right port name in the code, or don't put anything and let the program try to find your serial port.
3. You can also get connection error if you leave the Serial Monitor on. This is because the Arduino IDE is keeping the serial port busy. That is why closing the Arduino IDE after making sure it works is crucial.
4. If you have problems downloading Python or Pyserial, check Part 2 of the activity and do it again, or restart your computer to make sure your terminal/command prompt is up to date.

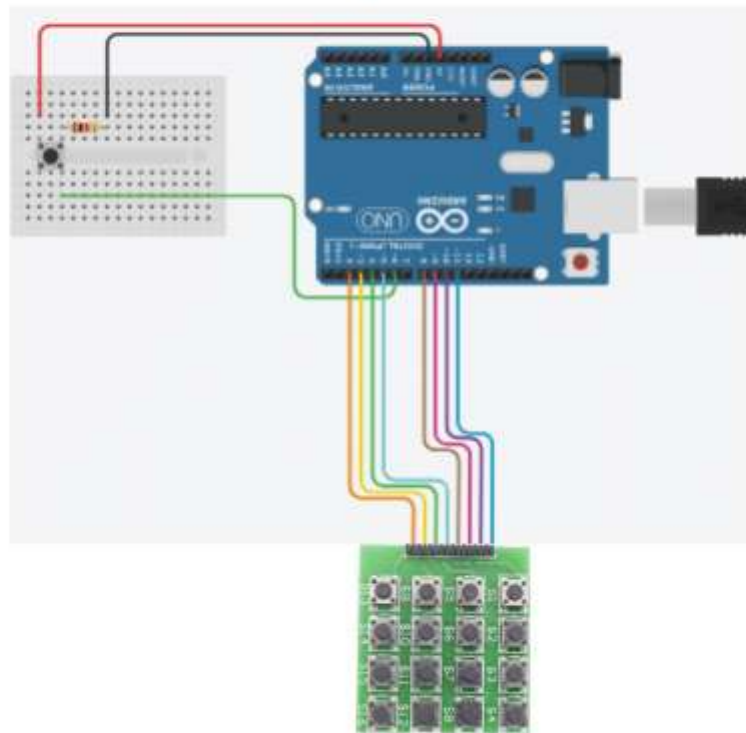
I. Challenge

Try adding more sensors to the Arduino board, or change the sensors we used in the activity, convert the readings you get from the sensors to useful units. Then use the Python program to store these data in your computer. You should change the name of the new file you are creating and using to store the reading in the Python code so that you don't overwrite the data you already collected. Simply look in the Python file where it says "output.txt" and change it to something else. It should still end with the extension ".txt".

Appendix N



INTRODUCTION TO MATRICES



A. Title of the Project

Introduction to Matrices

B. Subjects

Programming, Math

C. Knowledge

What you need to know

- How to use Arrays
- How to use loops
- How to use `digitalRead()`

What you will learn

- Matrices (array of arrays)
- Creating new functions
- How to use the 4X4 matrix keyboard
- Switch cases

D. Materials Needed



Jumper cables



1kΩ resistor



4x4 matrix keyboard



F-M wires



Pushbutton

E. Abstract

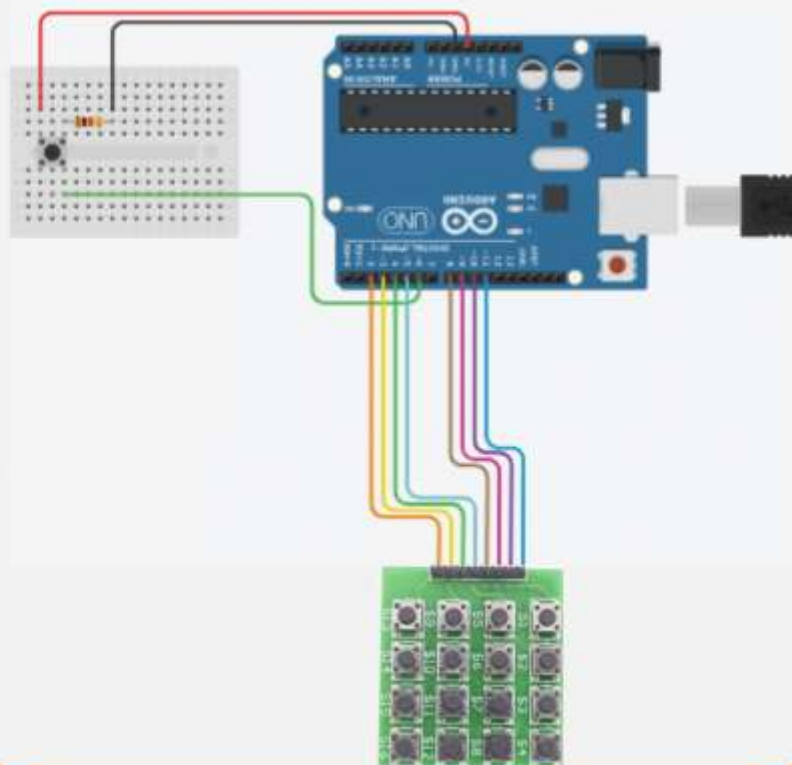
Matrices are an essential concept in many fields like math, computer science and physics, but what do they actually accomplish? Matrices help keep track of many dimensional measurements of a single entity. For example a two-dimensional vector has two values; its x and y values. However a three dimensional vector has one more value that is still comparable to its x and y value, which is the z value. Even though all these numbers are just numbers they can't be added to each other because they are not on the same plane. In this activity we will learn how to set up a matrix and how to make operations between two different matrices.

F. Context

You are asked to create a **matrix calculator**, which can add and subtract two matrices. The matrices have to be manually editable. The operation that will be used should be inputted from the serial monitor. Pressing a button on the matrix keyboard should increase the value of the element in that position of the matrix. You should be able to switch between the matrices by pressing the button on the breadboard. The operation will be done if you put plus "+" or minus "-" in the serial monitor and press enter.

G. Activity Description

G.1. Connection Diagram



The only thing to pay attention in this setup is the orientation of the 4x4 matrix keyboard. When the pins of the keyboard are above the buttons, the leftmost four pins control the rows, and the rightmost 4 pins control the columns. So the leftmost pin should be connected to pin 2 on the Arduino UNO, the pin next to it should be connected to pin 3 and so on. When you have the 4 leftmost pins connected to digital pins from 2 to 5, you should connect the other 4 pins to the digital pins from 8 to 11. Be sure that the rightmost pin is connected to pin 11. You should use female to male wires to connect the keyboard.

G.2. 4x4 Matrix Keyboard

The 4x4 matrix keyboard is basically a board with 16 push buttons on it. It is, in some sense, a predecessor to the 4x4 keypad. The big difference is that 4x4 keypad has numbers covering the buttons, whereas the keyboard is just buttons. In order to understand how the keyboard works, you need to understand rows and columns. A row is a horizontal line, a column is a vertical line. 4x4 means that there are four rows and four columns on this keyboard. There are eight pins for the keyboard, one for each row or column. When a button on the keyboard is pressed it completes a circuit and sends signal to the pins connected to the row and column of the button. It is important to debounce the key presses because when you are checking if a key is pressed in the loop() function, you might check it a thousand times in less than one second. This might create problems because the key would appear to be pressed many times, while it is just being pressed and held.

G.3. Matrices

Matrices and matrix operations are essential concepts of math, computer science and physics. A matrix is essentially an array of arrays. This means that every element in an array is another array. When talking about matrix sizes, we use two numbers. For example a 2x3 matrix would have two rows, and three columns:

$$A_{2 \times 3} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{bmatrix}$$

The elements in the matrix can be any positive or negative number (integer, fraction, percentage...). What do we use matrices for? We use matrices to do operations with objects/items with certain dimensions, and we want to keep all of these dimensions into consideration while we do the operations. For example if I have a matrix M where;

$$M_{2 \times 2} = \begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix}$$

and M multiplied by 2 is:

$$2M_{2 \times 2} = \begin{bmatrix} 4 & 2 \\ 6 & 2 \end{bmatrix}$$

So, every element in the matrix is multiplied by 2.

You can add and subtract matrices from each other. When you add or subtract matrices, you add or subtract the elements in the same positions. However this is only possible if the dimensions of the matrices are the same. For example if we have matrices A and B where;

$$A_{3 \times 2} = \begin{bmatrix} 2 & 3 \\ -1 & 5 \\ 5 & 1 \end{bmatrix}, B_{3 \times 2} = \begin{bmatrix} 1 & -1 \\ -1 & 9 \\ 3 & -2 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 3 & 2 \\ -2 & 14 \\ 8 & -1 \end{bmatrix}$$

However, if we had a matrix C where;

$$C_{2 \times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We cannot have A+C or B+C, because the dimensions do not match.

G.4. Program Code

```

/*
Credits to Chris Rouse and Cem Alemdar
Row 1 to Arduino pin 2
Row 2 to Arduino pin 3
Row 3 to Arduino pin 4
Row 4 to Arduino pin 5
Column A to Arduino pin 8
Column B to Arduino pin 9
Column C to Arduino pin 10
Column D to Arduino pin 11

Key identification:
(with connector at the top)
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
*/

//variables
int rowCounter = 0; // row counter
int columnCounter = 0; // column counter
int foundColumn = 0;
boolean foundCol = false;
int keyValue = 0;
int noKey = 0;
boolean readKey = false;
char operation;

```

```
int debounce = 400;

// pins
const int row1 = 2;
const int row2 = 3;
const int row3 = 4;
const int row4 = 5;
const int colA = 8;
const int colB = 9;
const int colC = 10;
const int colD = 11;
const int button = 6;

// matrices
int matrixNo = 0;
int matrix0[4][4];
int matrix1[4][4];
int result[4][4];
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(row1, OUTPUT);
  pinMode(row2, OUTPUT);
  pinMode(row3, OUTPUT);
  pinMode(row4, OUTPUT);
  pinMode(colA, INPUT_PULLUP);
  pinMode(colB, INPUT_PULLUP);
  pinMode(colC, INPUT_PULLUP);
  pinMode(colD, INPUT_PULLUP);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(noKey == 1){ // no keys were pressed
    readKey = true; // keyboard is ready to accept a new keypress
  }
  noKey = 0;
  if(digitalRead(button) == 1){
    Serial.println("");
  }
}
```

```

Serial.println("matrix changed");
Serial.println("");
matrixNo = !matrixNo;
delay(debounce);
}
for(rowCounter=row1; rowCounter<(row4 +1); rowCounter++){
  scanRow(); // switch on one row at a time
  for(columnCounter = colA; columnCounter <colD +1; columnCounter++){
    readColumn(); // read the switch pressed
    if (foundCol== true){
      keyValue =(rowCounter-row1) *4*(columnCounter - colA);
    }
  }
}
}
switch(matrixNo){
  case 0:
    if(readKey==true && noKey == 15){ // a key has been pressed
      //Serial.println(keyValue); // used for debug
      Serial.println("Matrix 1");
      updateMatrix0(keyValue);
      delay(debounce); // to wait after a key press
    }
    break;

  case 1:
    if(readKey==true && noKey == 15){ // a key has been pressed
      //Serial.println(keyValue); // used for debug
      Serial.println("Matrix 2");
      updateMatrix1(keyValue);
      delay(debounce); // to wait after a key press
    }
    break;

  default:
    break;
}
operation = Serial.read();
switch(operation){
  case '4':

```

```

Serial.println("Matrix 1 + Matrix 2");
matrixAdd();
break;

case '2':
Serial.println("Matrix 1 - Matrix 2");
matrixSubtract();
break;

default:
break;
}
}

void scanRow(){
for(int j = row1; j < (row4 + 1); j++){
digitalWrite(j, HIGH);
}
digitalWrite(rowCounter, LOW); // switch on one row
}

void readColumn(){
foundColumn = digitalRead(columnCounter);
if(foundColumn == 0){
foundCol = true;
}
else{
foundCol = false;
noKey = noKey + 1; // counter for number of empty columns
}
}

void updateMatrix0(int key){
int row = key / 4;
int column = key % 4;
matrix0[row][column]++;
for(int rowTotal = 0; rowTotal < 4; rowTotal++){
Serial.println("");
for(int columnTotal = 0; columnTotal < 4; columnTotal++){

```

```
Serial.print(matrix0[rowTotal][columnTotal]);
Serial.print(" ");
}
}
Serial.println("");
}

void updateMatrix1(int key){
int row = key / 4;
int column = key % 4;
matrix1[row][column]++;
for(int rowTotal = 0; rowTotal < 4; rowTotal++){
Serial.println("");
for(int columnTotal = 0; columnTotal < 4; columnTotal++){
Serial.print(matrix1[rowTotal][columnTotal]);
Serial.print(" ");
}
}
Serial.println("");
}

void matrixAdd(){
for(int i = 0; i < 4; i++){
Serial.println("");
for(int j = 0; j < 4; j++){
result[i][j] = matrix0[i][j] + matrix1[i][j];
Serial.print(result[i][j]);
Serial.print(" ");
}
}
Serial.println("");
}

void matrixSubtract(){
for(int i = 0; i < 4; i++){
Serial.println("");
for(int j = 0; j < 4; j++){
result[i][j] = matrix0[i][j] - matrix1[i][j];
Serial.print(result[i][j]);
```

```

        Serial.print(" ");
    }
}
Serial.println("");
}

```

G.5. Matrices in Programming

As it was mentioned before, a matrix basically an array of arrays. The way to initialize a matrix is as shown;

```
int A[ ][ ];
```

This initializes a matrix called "A" which has no set dimensions. You can initialize matrices with specific sizes by giving values in the square "[]" brackets;

```
int A[2][3];
```

This initializes a matrix with two rows and three columns. In order to place values into the matrix, or to get certain values in the matrix you have to call it as;

```
int value = 2;
```

```
A[0][0] = value;
```

Just like an array, the element numbers in matrices start from 0. This code stores the value 2 in the first element of the first row in the matrix. Just like arrays, it is useful to use for() loops to iterate through the matrix. However this time we are going to use a for() loop inside of a for() loop to iterate through the rows and columns. Here is an example of how we use this to print all of the elements in the matrix;

```

for(int rowTotal = 0; rowTotal < 4; rowTotal++){
    Serial.println("");
    for(int columnTotal = 0; columnTotal < 4; columnTotal++){
        Serial.print(matrix1 [rowTotal][columnTotal]);
        Serial.print(" ");
    }
}

```

The Serial.println() lines are used to visualize the matrices nicely in the serial monitor. The important thing to notice here is the **first for() loop** controls the rows, and the **second for() loop** controls the columns. This is because we want to print all of the elements in one row, then move on with the next row on a new line. In this piece of code, ever time the first for() loop runs, the second for() loop runs 4 times.

G.6. Switch Cases

The `switch()` function is a commonly used function, which is usually used instead of having many `if()` statements that check the same variable. How the `switch()` function works is as shown:

```
switch(variable){
  case 1:
    //what to do if the value of the variable is 1
    break;
  case 2:
    //what to do if the value of the variable is 2
    break;
  *
  *
  *
  default:
    //what to do if the value of the variable is none of the cases above
    break;
}
```

Switch cases are also useful if you are trying to only accept certain inputs. Here is an example of the same situation in the code:

```
operation = Serial.read();
switch(operation){
  case 1:
    Serial.println("Matrix 1 + Matrix 2");
    matrixAdd();
    break;
  case 2:
    Serial.println("Matrix 1 - Matrix 2");
    matrixSubtract();
    break;
  case 3:
    break;
}
```

```
}

```

In this code we are getting an input from the serial monitor about what **operation** to do with the matrices. We only want to accept two types of operations. Addition **+** or subtraction **-**. If we are given an input that is not + or -, the **default case** will run, which will not do anything.

G.7. Creating New Functions

So far you have only been using functions that were pre-written by someone else. However in this activity we have six new functions that we created. Let's remember what a function is. When a function is called, it does a small set of operations or tasks either on a given parameter, or on the program itself. For example, when we call the `Serial.println()` function and give a variable as a parameter, it will print out the value of that variable on the serial monitor. Creating your own functions is helpful because it cleans up the code, categorizes it, and makes it easier to read. Here is an example of a new function:

```
int increment(int var){
    var++;
    return var;
}

```

This is an example of a very basic function. When we call this function and pass in an integer as a parameter, it will increment the integer by 1, and **return** it. The word **"int"** before the function name indicates the data type of the variable that will be returned. In this case when we call the function we have to assign the value returned to a variable. For example:

```
int value = 4;
void setup(){
    value = increment(value);
}

```

There can also be functions where nothing is returned. In this case we use **void**.

So when the function is **called**, it will take the initial value of the variable, which is 4. Increment it by one, and return the incremented value. Then we store this incremented value in the variable "value".

```
int increment(int var){
    var++;
    return var;
}

```

Another important thing to notice is the **parameter** of the function. Basically, you have to initialize a variable in the parenthesis to act as a placeholder for what variable you want to pass in the function. Inside the function you have to do all of the operations on the variable you initialized, because when you pass in another variable in the **function call** the value of that variable will be assigned to the variable you initialized. This means that you have to **return** the variable you initialized in the parameter, because you are not

changing the value of the variable you pass in the function. You are changing the local variable you created in the function.

Here is an example of a new function in the activity's code:

```
void matrixAdd(){
  for(int i = 0; i < 4; i++){
    Serial.println("");
    for(int j = 0; j < 4; j++){
      result[i][j] = matrix0[i][j] + matrix1[i][j];
      Serial.print(result[i][j]);
      Serial.print(" ");
    }
  }
  Serial.println("");
}
```

This is a function that adds the two matrices we created in the activity. At the first glance we can see that this function will not return anything because it has `void` at the start. It also does not have any parameters because the two matrices that we add are global variables, so we can just use the actual variables. We also use another matrix called "result". This matrix is initially empty, but it is filled when you call the `matrixAdd()` function and then printed in the serial monitor.

H. What Could Go Wrong

1. Make sure that you copied the code correctly. It is reasonably long, and if you miss one curly bracket, your code will not run properly.
2. Be sure you connected the matrix keyboard exactly as shown in the diagram, because the order of the cables matter a lot.
3. Make sure the button is connected using a resistor, and as a digital input, because we only need information about if it is pressed or not.

I. Challenge

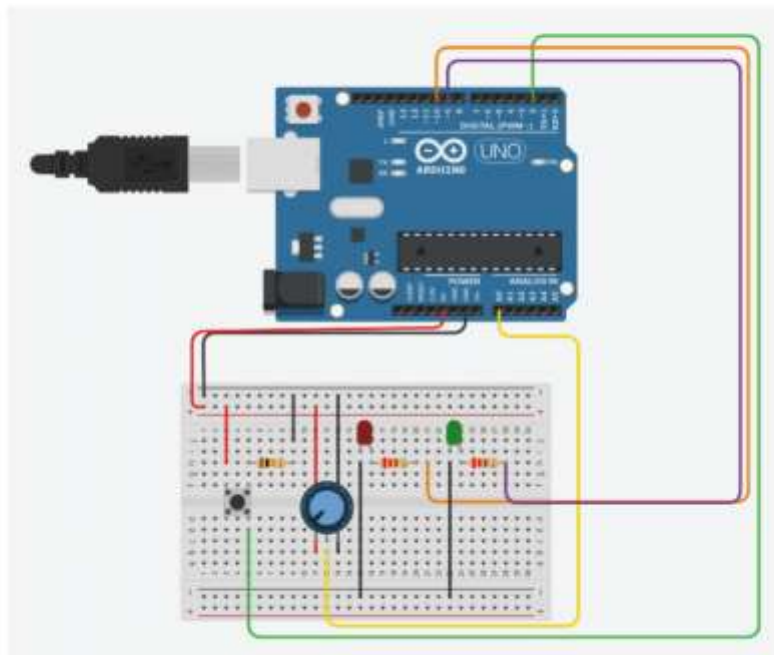
Use the keyboard matrix, buttons and a buzzer to make a small synthesizer. A good way to approach this is keeping a matrix of frequency values. When you press a button on the matrix keyboard, the buzzer will play the corresponding frequency value in the matrix. You can also have one or more buttons to change the tempo or the frequency of the sound (multiply, divide, oscillate...). Look into the `tone()` function on the internet. Be creative, you can also put frequency values of real notes and create an instrument.

Appendix O



MAKING A DIAL LOCK

[Arduino STEAM]



A. Title of the Project

Making A Dial Lock

B. Subjects

Programming, Electrical Engineering

C. Knowledge

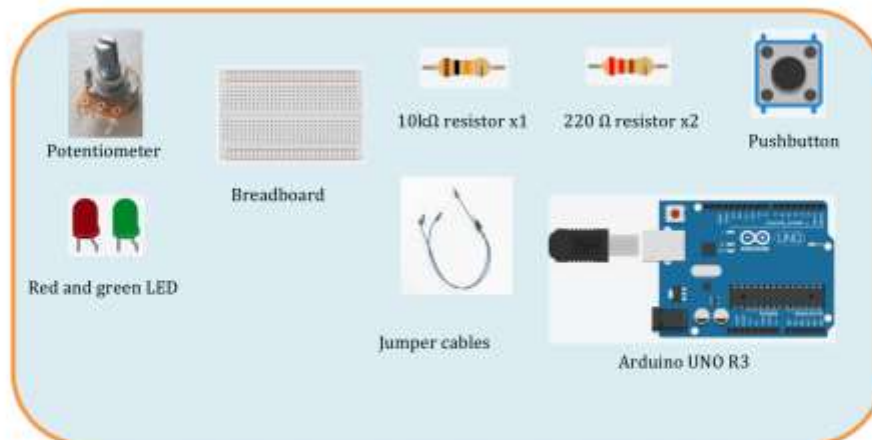
What you need to know

- What is a loop
- How to get digital and analog input
- How to use the Serial Monitor
- How to increment variables
- How to use read() and write() functions

What you will learn

- How to use for() and while() loops
- How to use arrays to store information
- How to compare two arrays
- What a potentiometer does in a circuit.
- How to use a push button

D. Materials Needed



E. Abstract

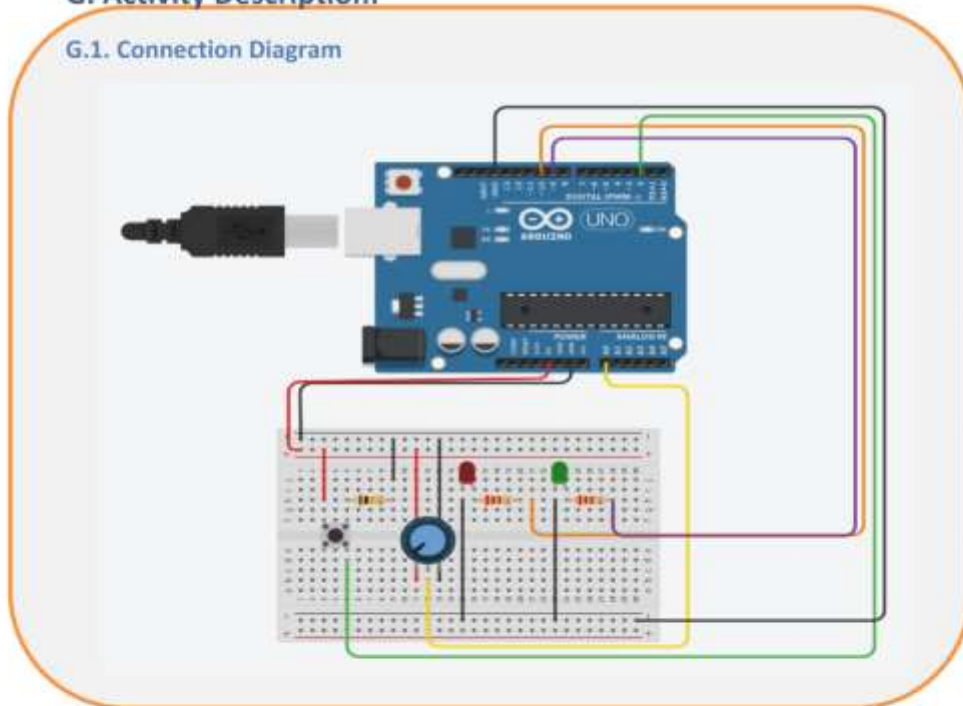
In this workshop you will try to simulate a locking mechanism for an electronic briefcase. The potentiometer will be the dial and we will use a red LED to indicate it is locked and a green LED to indicate it is unlocked. You will use the push button to enter the number that the dial is showing. You will also use programming concepts like while loops, and arrays. When you complete this workshop you will hopefully learn how to get analog data, compare it to digital data, and how to store this data.

F. Context:

You are asked to design a security system for a briefcase. This security system has to be more secure than a pin pad, which uses a 4-digit passcode. It should have an indicator to let the user know if the briefcase is locked or unlocked.

G. Activity Description:

G.1. Connection Diagram



There are few things that are important to notice about the diagram. First of all the middle pin of potentiometer is connected into pin A0, which is an analog input pin. So when we call the `analogRead()` function it will give us a value between 0-1023. We will use the potentiometer as a dial. However if we keep the values at 0-1023 it will be difficult to get specific numbers we want because it is very sensitive. A slight touch can change the value by 50. Therefore, we are going to scale this where the max value will be 100. This will still be more secure than a 4 digit code because we will have 4 numbers between 0-100 as our combination. Another detail is that the button is connected to pin

2, which is a digital pin, because the value we want from the button is 1 for pressed and 0 for not pressed. We will be using `digitalRead()` to gather this input.

G.2. Passcode Security

Authentication has been extremely important with the rapid growth of the technology industry. The most common way to authenticate the user is passcode or password. These passcodes are generally 4 or 6-digit numbers. Which means that every digit is a number between 0-9. This creates a total of ten thousand different combinations for 4-digit and one million combinations for 6-digit. However with the technological advancements, computers have become incredibly powerful as well. Which means that a well-written computer program can try thousands of combinations in less than a second. Of course all of the devices that use passcodes have other security measures like locking itself if there are too many wrong attempts or making them wait for a while until they can try again. Right now most of the phones we use have changed from passcodes and passwords to biometric authentication such as fingerprint or face recognition. These are better ways of authentication simply because everyone has a unique fingerprint and face (this may change for twins). They are also incredibly difficult to recreate in real life.

G.3. Potentiometer:

A potentiometer is a type of variable resistor that has three terminals. The outer ones should be connected to the circuit, and the middle terminal should be connected to an analog input. The middle terminal is attached to a sliding contact, called a wiper, that moves over the resistive element in the potentiometer. The resistive element can be seen as two resistors connected in series, and the wiper adjusts the ratio of the first resistor to the second resistor. It does not matter which side you connect the anode or cathode of the circuit, unlike LED's, it just reverses the wiper direction.



G.4. Push Button:

The push button is a pretty simple but useful part. It can function as a switch or a button. If you use only two terminals the button will act as a switch. When it is pressed the circuit is closed and current can flow through, but when you release it, the circuit will be open. So instead of having to keep it pressed, we can connect another terminal of the button to one of the digital pins on the Arduino board. This allows us to use the `digitalRead()` function to see if the button is pressed or not. Since this is a digital input, the value will either be 1 or 0, where 1 means that the button is pressed. To keep track if a button has been pressed or not we use a type of variable called "flag".

```
int buttonState;
if(digitalRead(button) == 1){
    buttonState = 1;
}
else{
    buttonState = 0;
}
```



In the given example above, the variable `buttonState` is a flag because it keeps track if the button has been pressed or not. You can also use the same concept as an On/Off button.

```
int buttonState = 0;
if(digitalRead(button) == 1){
    buttonState = !buttonState;
}
```

In this case the state of the button is reversed every time the button is pressed. However this time we had to give the flag an initial value because putting `!` before a variable means "not". So not 0 is equal to 1, and not 1 is equal to 0.

G.5. Program Code:

```
// pin numbers
int ledRed = 10;
int ledGreen = 9;
int poten = A0;
int button = 2;

// other variables
int reading; //analog reading
int conversion; //converted analog reading
int buttonState = 0; //the button is pressed or not
int combination[4] = {15, 74, 3, 47}; //the correct combination (you
can change it)
int entered[4]; //the numbers that are entered

void setup() {
    pinMode(ledRed, OUTPUT);
    pinMode(ledGreen, OUTPUT);
    pinMode(poten, INPUT);
    pinMode(button, INPUT);
    Serial.begin(9600);

    analogWrite(ledRed, HIGH);

    int num = 0; //the count of numbers entered
    while(num < 4){ //this loop will run until 4 numbers are entered

        reading = analogRead(poten);
        conversion = map(reading, 0, 1023, 0, 100);
        Serial.println(conversion); //to see which value you are on the
potentiometer
        delay(100);

        if(digitalRead(button)){ //check if the button is pressed
            buttonState = 1; //1 means it is pressed 0 means it is not
            entered[num] = conversion; //appending the value into the
entered numbers array
            num++;
            Serial.println("number entered");
            delay(1000);
        }
    }
}
```



```

    }
}

int correct = 0; //number of values in the entered array that match
the combination
for(int i = 0; i < 4; i++){ // loops 4 times

    if(combination[i] == entered[i]){ //check if the entered value
matches the combination
        Serial.println(entered[i]);
        correct++;
    }
    else{
        num = 0;
        Serial.println("Wrong combination");
        break; // get out of the for() loop
    }

    if(correct == 4){ //check if all values were correct
        Serial.println("UNLOCKED");
        analogWrite(ledRed, LOW);
        analogWrite(ledGreen, HIGH);
    }
}
}
}
void loop() {
    //There is nothing in the loop
}
}

```

G.6. Loops:

There are two types of loops that are essential in every program. `For()` loops and `while()` loops. These are programming concepts that apply to almost every programming language, unlike the `loop()` function in Arduino, which is special for microcontrollers.

The main difference between a `for()` and a `while()` loop is; `for()` loops, loop for a certain number of times and then stops. Where a `while()` loop waits for an argument to be false to exit the loop. If the argument never becomes false it will go on until you manually stop the program or run out of memory. This is called an infinite loop, which usually causes problems. But there are also useful infinite loops, such as the `loop()` function in Arduino. Here are some examples of loops and how they are structured:

```

int count = 0;
for(int i = 0; i < 5; i++){
    count++;
    Serial.println(count);
}

```

This piece of code is a simple use of a `for()` loop. We have a variable outside of the `for` loop that is called "count", which will keep count of how many times we go through the loop. Inside the brackets of the `for()` loop we initialize a variable `i` and assign the value 0. Then we indicate how long the `for` loop will run for. In this case we want it

to run until variable `i` is less than 5. This means that the loop will run for 5 times because `i` starts at 0 and we add 1 to it when each loop ends. Variable `i` is incremented by 1 each time because we put `i++` for the third parameter of the `for()` loop. It is important that you put a semicolon ";" after each parameter in the brackets. The inside of the `for()` loop is the part enclosed by the curly brackets "{}". Every time the loop runs, it starts from the top of the loop and runs each line of code, then loops back to the top and does it again.

```
int count = 0;
while(count < 5){
    count++;
    Serial.println(count);
}
```

This is a simple example of a `while()` loop. This piece of code does exactly the same thing as the `for()` loop. The important thing here to notice is that there is only one parameter inside the brackets of the `while` loop. It is called a **statement** because it either has to be true or false. The variable `count` is an integer, so if its value is less than 5 when the loop comes back to the top it will run the loop again until that statement becomes false. Which will be after looping 5 times.

A very simple example of an infinite while loop would be:

```
while(1){
    // loop body
}
```

Since the statement will always be true because the statement is 1, which means true, and 0 is false. In these kinds of loops there is one way to exit the loop, which is the function "break". You can see an example use of `break` in the code of the activity. It is not being used to exit an infinite loop; however, it is used to exit the loop when any of the entered numbers are wrong. Because we don't have to check the rest of the numbers if one number is wrong. Basically what `break` does is; it leaves the loop where the function is called, but the program keeps running the code outside the loop. Here is the example:

```
for(int i = 0; i < 4; i++){ // loops 4 times
    if(combination[i] == entered[i]){ // to check if the entered value
                                                //matches the
combination
        Serial.println(entered[i]);
        correct++;
    }
    else{
        num = 0;
        Serial.println("Wrong combination");
        break; // get out of the for() loop
    }
}
```

G.7. Arrays:

Arrays are a data type that can store many values of the same type. They are a kind of list where each element in the list has a specific position. For example an array can be written as:

```
int nums[5] = {2, 4, 6, 1, 8};
```

This is an array called "nums" and it can have at most 5 elements in it, which is assigned by the number in the square brackets "[]". In order to get one of the values in an array you have to call for it like `nums[2]`. Now here is the tricky part. When you call `nums[2]` it returns you the third element in the array, which is 6. This is because the first element in the array is represented as `nums[0]`.

You can't assign an array to a variable. For example:

```
int number = 0;
int nums[5] = {2, 4, 6, 1, 8};
number = nums;
```

is NOT possible. However,

```
int number = 0;
int nums[5] = {2, 4, 6, 1, 8};
number = nums[1];
```

is possible. It also works vice versa:

```
nums[1] = number;
```

is also possible. But they don't do the same thing. The first one assigns the value stored in `nums[1]` to the variable `number`. The second one assigns the value of the variable `number` to the second element of `nums`, which is `nums[1]`.

You can also have empty arrays and fill them up in the code. There is an example again in the code for the activity. The array "entered" is empty when initialized, but it gets filled up as the user enters numbers. The first number is stored in `entered[0]`, second number in `entered[1]`, and so on.

When it comes to comparing arrays it is not as simple as:

```
array1[] == array2[];
```

In order to compare if two arrays are the same or not we have to write a loop that compares each element in the array one by one. Here is how we do it in the activity:

```
for(int i = 0; i < 4; i++){ // loops 4 times
    if(combination[i] == entered[i]){ // to check if the entered value
                                                // matches the
combination
        Serial.println(entered[i]);
        correct++;
    }
    else{
        num = 0;
        Serial.println("Wrong combination");
        break; // get out of the for() loop
    }
}

if(correct == 4){ // to check if all values were correct
```

```

    Serial.println("UNLOCKED");
    analogWrite(ledRed, LOW);
    analogWrite(ledGreen, HIGH);
  }
}

```

The most important thing to notice in this `for()` loop is the variable `i`. This variable both keeps track of the number of times the loop runs for, and the element number in the array. The first time the loop runs, variable `i` is 0. So in the `if()` statement, we check if `combination[0]` is equal to `entered[0]`, which are the first elements of the arrays. Then the second time the loop runs, variable `i` will be equal to 1 so it will check the second elements in the arrays, and so on.

H. What Could Go Wrong

1. Make sure the short legs of the LED's are connected to the ground pin and the long legs to a resistor and then to a digital pin. This is important because if you connect it in reverse the LED will not work.
2. Be sure to connect the middle pin of the potentiometer to the "A0" pin, because we want to collect analog input from it.
3. Be sure to use a 10K Ω resistor for the push button and connect it to a digital pin, because we want to collect digital input from it.
4. If some parts are not working and you don't know why, try changing the cables connected to the parts that are not working. The jumper cables can be very unreliable.

I. Challenges:

I.1. Mechanical Challenge:

Try to add a buzzer to the setup you have, which will go off if you input a wrong combination. Use the `tone()` function to make the buzzer create a sound. Be creative with your alarm tone. You can use loops to repeat the alarm or change the frequency. Here is an example use of the `tone()` function:

```

int buzzerPin = 4;
tone(buzzerPin, 2000, 500);

```

The parameters in the `tone()` function are:

```

tone(pin number, frequency in Hz, duration in ms)

```

I.2. Code Challenge:

Try adding a new way to change the combination without changing the Arduino setup. Feel free to change the code however you want. There are many different ways to approach this while keeping the security high. For example making it possible to change the combination while the system is unlocked is one way. Having a "secret" code that you can put in to change the combination is another way... Use all the information you learned from this activity to come up with your own way to do it. Use loops, arrays, array comparisons etc.