Microcontroller Lesson Manual for Teachers and Students Alike to Improve Understanding of Electricity and Circuits through Hands-On Application

Caitlin Mastroianni

A capstone submitted to the faculty of
Brigham Young University
In partial fulfillment of the requirements for the degree of

Bachelor of Science

Adam Bennion, Advisor

Department of Physics and Astronomy

Brigham Young University

April 19, 2023

ABSTRACT

Microcontroller Lesson Manual for Teachers and Students Alike to Improve Understanding of Electricity and Circuits through Hands-On Application

Caitlin Mastroianni
Department of Physics and Astronomy, BYU
Bachelor of Science

Education standards have evolved rapidly within the 21st century, transitioning from content-based lectures and memorization to hands-on learning which encourages creativity, problem-solving, and cognitive engagement. Through the implementation of low-cost microcontrollers, such as Arduinos, in the classroom, teachers and students alike are encouraged to explore the concepts of electricity and circuitry through experimental learning. A lesson manual, intended to be used as introductory material for microcontrollers, was created in the hopes to fulfill and exceed secondary education standards/requirements, specifically within the state of Utah. Said manual was revised to a completed and presentable state with the assistance and feedback from BYU students within the Department of Physics and Astronomy who were in pursuit of their bachelor's degree in Physics Education.

Keywords: [Microcontroller, Arduino, Lesson Manual, Secondary Education, Physics Education, Hands-on Learning, Active Learning]

ACKNOWLEDGMENTS

Primarily, I'd like to thank Professor Adam Bennion for helping guide me throughout this capstone and offering me the opportunity to see my work in effect. I'd also like to thank Professor Merrell for allowing me to work with his students within his classroom and financially aiding with the collection of required electrical components for his students.

I would like to thank my parents for devoting their lives to giving me every opportunity imaginable. Thank you for the sacrifices you've made to come to America to offer me the American dream along with the outstanding example of persistence, patience, and love of family. Thank you for also making me apply to BYU, just in case I decided to transfer from ASU - it happened to work out quite wonderfully.

I would like to thank my husband, Spencer, for being by my side through thick and thin. For being my laughter, my strength, and my shoulder to cry on. You have kept me going through the times when I thought I couldn't do it anymore and have carried me to my finish line.

Finally, I would like to thank my service dog, Winston. Thank you for staying by my side through the long nights, for offering me kisses when I truly needed them, and for the endless days of dedicated work.

Table of Contents

Introduction

1.1 Background

1.1.1 Active Learning in STEM Classrooms

Scientific education can be divided into two broad domains: content knowledge and processing skills. (Loewenberg Ball, 2018). Content or declarative knowledge is defined to be the realm of understanding and memory such as statistics, principles, conceptual models, and fact-based theory. Processing or procedural knowledge, such as observation, measurement, and the ability to progress through the scientific process, expands the declarative knowledge of students (Hurca 2013). Recent research has generated support for a transition from teacher-centered instruction to learner-centered, active learning which expands a student's ability to develop both declarative and procedural knowledge. (Freeman et al. 2014).

"A child best learns to swim by getting into water; likewise, a child best learns science by doing science" (Rillero, 1994). It was determined that students struggling to understand concepts explored and taught in a secondary-level physics classroom could be categorized into two main groups: students who were not familiar with the subject from daily life" and "students who couldn't embody abstract concepts". (Aycan & Yumuşak, 2002; Karakuyu, 2008). The student-centered teaching method requires educators to remain facilitators and guides whilst the focus shifts onto students, encouraging their engagement through problem-solving, communication, collaboration, and scientific inquiry. Reasoning skills, the ability to perform independent critical thinking, and a deep understanding of complex topics developed from constructivist teaching styles encourage interpersonal skills that lead to open-ended, complex, and team-based exploration within the boundaries of scientific experimentation.

The Department of Psychology's Human Performance Lab, directed by Professor Sian Beilock, published a study in 2015 that provides both a statistical and biological insight into the benefits of constructivist teaching styles within a physics classroom. The study determined that action experience (relative to observation) led to increased activation of sensorimotor systems – up to 57.9% more stimulation – important for representing physical concepts. Activation, in turn, enhanced the students' understanding of torque and angular momentum. (Kontra et al. 2015)

Utah State Board of Education (USBE) established a policy requiring the identification of specific core standards to be met by all K-12 students to graduate from secondary education. Utah Science with Engineering Education (SEEd) declares that a principle of science learning is to be personal and engaging. Their definition of this criteria is as follows:

> Research in science education supports the assertion that students at all levels learn most when they can construct and reflect upon their ideas, both by themselves and in collaboration with others. Learning is not merely an act of retaining information but creating ideas informed by evidence and linked to previous ideas and experiences. Therefore, the most productive learning settings engage students in authentic experiences with natural phenomena or problems to be solved (*USBE- Utah State Board of Education*).

When teachers were asked which topic among the various constructs in the physics curriculum was the most challenging for students in terms of conceptual learning, results showed that current, voltage, and power in electrical circuits were the most difficult (Aykutlu et al. 2015). Furthermore, Strand PHY.2: Energy, under the Standard PHYS.2.4 states that students are required to:

> Design a solution by constructing a device that converts one form of energy into another form of energy to solve a complex real-life problem. Define the problem, identify criteria and constraints, develop possible solutions using models, analyze data to make improvements from iteratively testing solutions, and optimize a solution. Examples of energy transformation could include electrical energy to mechanical energy, mechanical energy to electrical energy, or electromagnetic radiation to thermal energy. (PS3.A, PS3.B, ETS1.A, ETS1.B, ETS1.C) (*USBE-Utah State Board of Education*).

Combining the beneficial effects of hands-on learning and the means of increasing achievements surrounding electricity, microcontrollers could offer a cost-effective and hands-on route to teach students the concepts of electricity and energy transformation (Slugan et al. 2017).

1.1.2 Implementations of Microcontrollers in the Classroom

A microcontroller unit, or MCU for short, is a single integrated circuit comprised of various elements. Elements found on MCUs typically include a microprocessor, input/output ports (I/O), random access memory (RAM), read-only memory (ROM), timers, and counters accessed by varying ports and pins on the unit itself. Arduino is an open-source electronics platform based on easy-to-use hardware and software. The simplistic design of the single-board microcontroller, along with microcontroller kits, allows for the easy building of digital devices and is widely available, and a low-cost option.

In what can be described as a micro-laboratory, Arduinos naturally lends itself to introducing students to programming, automation, and robotics (Marzoli 1970). The main aims of implementation of micro-controllers, including Arduinos, within the classroom, are: ": 1)

Encourage interest in microcomputer concepts and understanding of measurements, natural phenomena, and concepts. 2) Develop critical-logical and systematic thinking. 3) Develop experimental skills through the study of the deterministic nature of physical laws. 4) Formulate research questions and hypotheses. 5) Systematize and analyze data." (Slugan et al. 2017).

Ateş and Eryilmaz examined the effectiveness of hands-on and minds-on activities with ninth-grade students' achievements and attitudes toward simple electric circuits. Students split up into two groups – a control group who learned through lecture and an experimental group who engaged in an activity – were then taught an electrical concept and took a pretest and a posttest. In the case of achievements, a noticeable change of 22% in the pretest and posttest was found in the experimental group when compared to the 2% change in the control. (ATEŞ et al. 2011). The implementation has proven beneficial effects on academic achievements and increased students' motivation and creativity when applying the microcontroller concept to real-life applications (Santosa et al. 2019).

In the hopes of fostering individual curiosity and interest in exploring natural phenomena, curricula such as these encourage students to find the applications of concepts in everyday life, therefore, increasing the fun, creative, and exploratory nature of physics. With the design of circuits within a microcontroller, foundational layers for software design are laid with simple objectives such as input/output interactions of lights, motors, and sensors inspiring the next generation of STEM students (Litts et al. 2017).

1.2 Purpose of Study

Whilst completing my undergraduate degree in Physics, a course introducing skills to foster innovation and leadership in experimental science through hands-on experience introduced Arduinos to explore circuitry and electrical theorems, such as Ohm's law, in practical applications. After taking this course, I found great interest in the capabilities and opportunities that an Arduino presented. I spent my free time at home creating various circuits that performed many small tasks as well as creating my own, simple "R2-D2" robot that was able to move independently, perform a tiny dance while lighting up, and play the robot's classic beeping.

In the summer of 2021, I was granted the opportunity to apply my newly gained knowledge by working alongside Doctor Nathan Powers at Brigham Young University as a teaching assistant for a summer course that high school teachers in nearby districts attended; the course aimed to impart new teaching applications through laboratory and hands-on methods. As a teaching assistant, my responsibilities were to help answer general questions while walking through various experiments and learning segments as well as creating projects involving Arduinos that encouraged discovery and aided students (and teachers) in their application of electricity. Additionally, I helped create motherboards and add-ons for Arduinos with capabilities such as photoresistors and SD recorders to measure the velocity of an object.

Working with the teachers during the week-long course, I noticed a disconnect when we first introduced Arduinos. Due to the teachers' lack of knowledge and experience with Arduinos, they were taught how to interact with the boards by following the "basics" tutorials provided on the Arduino website. Although the lessons are rudimentary, they lacked essentials such as clear instructions, provided code with explanation, and encouragement to understand circuit components. Teachers approached the microcontroller as a task to be completed, throwing wires

and components into the mix with the hopes of "just making it work" without understanding what they were creating. After noticing the way that teachers approached learning this type of circuitry, it was likely that students would mirror the behavior.

The purpose of this capstone is to provide a framed structure through lessons in which both teachers and students can learn and apply principles of electricity and circuitry in a physical space. It becomes apparent that preservice teachers in secondary education, most specifically physics instructors, should prepare syllabuses that build from low-cost materials and experiences in which undergraduates can enhance their student's scientific process skills through personal discovery and experimentation (Hurca .N. 2012).

This capstone aims to provide students and future teachers with a foundational knowledge of microcontrollers through the physical implementation of electrical concepts. As the undergraduates work through the created lessons, they can gain confidence by exploring circuits whilst creating a connection between models such as electricity and classic mechanics. The medium will be Arduinos - Arduino is an open-source electronics platform based on easy-to-use hardware and software. The simplistic design of the single-board microcontroller and microcontroller kits allows for the easy building of digital devices and is widely available, and a low-cost option.

Methods

2.1 Manual Structure

The creation of the manual/workbook will be founded on previous coding projects on which I've established my knowledge of microcontrollers. The workbook will be created under the assumption that teachers or students are interacting with microcontrollers for the first time and will commence with introductory material that will both introduce the benefits of Arduino use in the classroom along with the basics of the Arduino unit itself. In the hopes of exceeding the standard provided in the basic tutorials provided by Arduino, the manual will focus on providing introductory information about all components of the microcontroller and equipment, as well as detailed breakdowns of circuits and code.

The manual will be separated into 4 parts as detailed below:

Introduction

The introduction will include an explanation of the manual's purpose through a collection of background information on the importance of hands-on learning and an explanation of how microcontrollers apply to the manual's main objective.

Microcontroller Basics

Microcontroller basics will include a breakdown of the basic knowledge necessary to begin working through the lessons. The topics covered will be the Arduino integrated development environment (IDE), Arduino IDE coding essentials, Arduino schematics, and circuitry basics.

Lessons

The lessons contained in the manual will cover introductory circuits and coding with each consecutive lesson building on the previous lesson's principles. Lessons will take students step-by-step through each lesson's circuits, giving them a foundation of how to create a circuit with an objective in mind (such as lesson one's objective being to blink an LED). The lessons aim to make students comfortable with the material enough to create their circuits, such as those encouraged in the lesson's exploration section, as well as understand electrical material and phenomena better.

Project

The final project is created as a cumulative experiment in which students are to complete a goal using the knowledge and principles explored in the lessons.

2.2 Organization of the Subject Curriculum

2.2.1 Microcontroller Basics

As described above, the first section of content will cover the basics of microcontrollers from coding to circuitry essentials. This section aims to give students a basis of understanding of the components they will be working with throughout the lessons and to encourage students to use their knowledge to problem-solve independently. The individual segments expanded upon are:

Arduino Integrated Development Environment (IDE)

The Arduino microcontroller is a physical programmable circuit board that, combined with a piece of software or an integrated development environment, can be coded to perform different tasks by controlling the board's output/input capabilities. The IDE contains a text editor for writing code, a text console, and a toolbar with buttons for a series of functions and menus. Programs written using Arduino IDE are called sketches, which within the software can be compiled, debugged, and uploaded to the board. Going through the basic elements of the IDE a student can gain a basic understanding of how to work the IDE software in connection to the board.

Arduino IDE Coding Essentials

The Arduino IDE software is based on a variant of the C++ programming language. A simplified version of the C++ language, along with simplified syntax and pre-defined functions, is used because it is a language simple enough for inexperienced coders to compile a functional program. Students are introduced to basic pre-defined functions that are frequently used to create a practical sketch – this section covers both the basic function's structure, description of the function's purpose, and example implementations.

Arduino Schematics

Arduino boards are microcontrollers based on the ATmega328P, with 14 digital input/output pins (with various pins capable of PWN outputs), 6 analog input pins, a 16 MHz resonator, a power jack, a USB connection, and a reset button. In an included diagram, the Arduino board is detailed piece by piece with each pin's ability described, and should be used as a reference for students whilst building circuits and connecting them to the board.

Circuitry Basics

Circuit essentials such as explanations of the use of breadboards, circuit symbols, and component descriptions are included in this section. When an electric current is provided, such as through an Arduino board, within an electrical circuit, electrical energy can be transformed into other forms of energy that do work such as through light in LEDs, rotation of motors, and can be manipulated through elements such as potentiometer and buttons. For students to understand what they will be building, students must comprehend every piece of a circuit and how they play a role in a circuit's function.

2.2.2 Lesson Structure

The lessons contained in the manual will cover introductory circuits and coding with each consecutive lesson building on the previous lesson's principles. Lessons have been created to cover basic circuits with each lesson split into parts allowing those working through it to understand each task completely whilst also encouraging personal exploration of circuitry and coding. The lessons are structured in parts as follows:

Hardware Required

A list of the circuit components necessary to complete a lesson is included at the beginning to allow both teachers and students to gather everything necessary at the outset.

Schematic

A schematic of the electrical circuit needed to be built to apply properly functioning code is presented at the beginning of the lesson. This section includes not only a symbolic

representation of the circuit but also includes a basic explanation of how the circuit can be connected.

Step-By-Step Programming

After students have completed their circuit, a step-by-step guide takes them through the coding portion with descriptions of what each line of code means. Whilst working through the program, students are encouraged not only to understand the intention behind the code but begin to understand the functionality of circuit components when interacting with the board.

Completed Code with Comments

A screenshot of the Arduino IDE with the completed code, with comments explaining a code's line meaning, is included at the end to give students a complete view of the compilation and completed product of the step-by-step guide.

Exploration

Most lessons include a section called exploration that encourages students to answer questions founded on the knowledge that they gained through every lesson. This section's purpose is to encourage students to manipulate and "play" with circuits, and to provide a space where students are allowed to explore, experiment, and create an independent understanding of electricity in electrical circuits.

2.2.3 Project Structure

The manual ends with a final project that explores the principle of electrical energy being converted into mechanical energy while utilizing the knowledge gained throughout the lessons. This section aims to give students a "problem" to solve through which a student's curiosity and

interest in exploring natural phenomena encourages them to find the applications of concepts in everyday life increasing the fun, creative, and exploratory nature of physics. Teachers are allowed to implement this section within their classrooms in a competitive nature if wanted, to encourage students to push their project's limits whilst working with their peers.

2.3 Implementation of Lesson Manual within a Classroom

A completed, comprehensive draft of the tutorial manual was given to undergraduates at Brigham Young University in the Physics Education Major in the PHSCS 311 class, overseen by Professor Duane Merrell; this opportunity was made possible by my advisor, Professor Bennion. The students worked through the manual providing feedback on the work they've completed in correlation to how the workbook has helped and can be improved to better help understand the basics of electricity and circuitry. This application within a classroom setting between the capstone manual and the students of PHSCS 311 took place in scheduled classes on November 30th and December 5th of 2022.

Results and Discussion

3.1 Classroom Implementation Observations

On November 30th and December 5th of 2022, undergraduates at Brigham Young University in the Physics Education Major worked through the tutorial manual under the supervision of Professor Duane Merrell, Professor Adam Bennion, and myself to collect as much data and feedback as possible. While working alongside them, any questions or comments made were noted and talked through for later review when revising the tutorial manual for a final draft.

Initial Impressions were generally positive. I often checked in with them about their feelings, questions, or changes that they would like to see to clarify the material or make it more engaging for students. Personal observations were also noted about how the students interacted with the manual were referred to when revising the manual.

3.2 Feedback and Consequential Corrections

Below is a gathered list of comments and notes that were made throughout the classroom observation period and consequential corrections that were made to the manual within the final draft presented.

3.2.1 Original Manual Structure

Student Note: "It clearly defines what you will find in each section… teachers will have an easier time leading students to the proper place"

The original main structure of the manual remained unchanged.

Original and Revised Structure: Introduction, Microcontroller Basics, Lessons, Project

Observations within the Microcontroller Basics Section

When observing the students working through the primary draft of the manual, I noticed there to be a lack of basic knowledge specifically concerning electric circuits and their elements. Students often remarked on the lack of understanding between each element's purpose, structure, and implementation within the circuit itself. The questions students asked were questions such as "What is a breadboard? How does this work? How do I put in the elements in the breadboard? How does a potentiometer work? What do the colors on a resistor mean?". Therefore, additional sections and subsections were added to microcontroller basics.

Original Microcontroller Basics Structure

Arduino Integrated Development Environment

Opening Page, IDE Buttons, and Setup

Most Frequently Used Functions

Example Implementations of Frequent Functions (DIGITAL)

Example Implementations of Frequent Functions (ANALOG)

Arduino Schematics

Revised Microcontroller Basics Structure

Arduino Integrated Development Environment

IDE Button Functions

IDE Coding Setup and Loop

Output Window

Arduino IDE Coding Essentials

Most Frequently Used Functions

Example Implementations of Frequent Functions (DIGITAL)

Example Implementations of Frequent Functions (ANALOG)

Arduino Schematics

Circuitry Basics

Breadboard Diagram

Circuit Symbols

Component Explanations

3.2.2 Introduction

No comments noted.

3.2.3 Microcontroller Basics

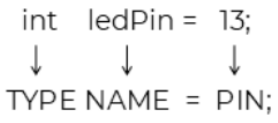
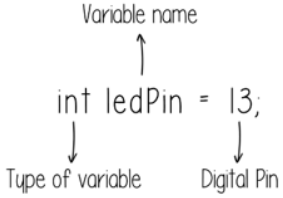
Student Note: "Breaking down the Arduino IDE is a little difficult not knowing what to expect when you first download the software"

The Arduino IDE section was revised to include more specific sections about the individual parts of the software instead of lumping them all together under one header. The more detailed descriptions aim to help guide students about each piece of software before working with it.

Original IDE Sections	Revised IDE Sections
- Opening Page, Buttons and Setup	- IDE Button Functions - IDE Coding Setup and Loop - Output Window

Student: "The coding basics section seems a little "wordy", the students might have a hard time reading through it all"

Although, understandably, the section itself is word-heavy, it is still necessary to include descriptions of common functions. However, to cater to the idea that words alone on the page would be difficult to grab students' attention, the revised manual has a more eye-catching style. The descriptions of the functions were also revised to be as minimal and concise as possible without detracting from their purpose.

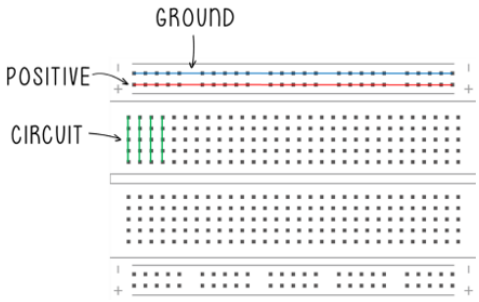
Original Function Description Image	Revised Function Description Image
	

3.2.4 Lessons

Note: Students asking about where the circuit goes, maybe add a breadboard diagram?

The updated circuitry basics section now includes a diagram along with a very thorough explanation of what a breadboard is and how it's used to build electrical circuitry. This was included to make it easier for students to comprehend the components of their kits and how each piece functions so that they are acquainted with what they are interacting with and building with when they arrive at the lessons.

Revised Breadboard Diagram

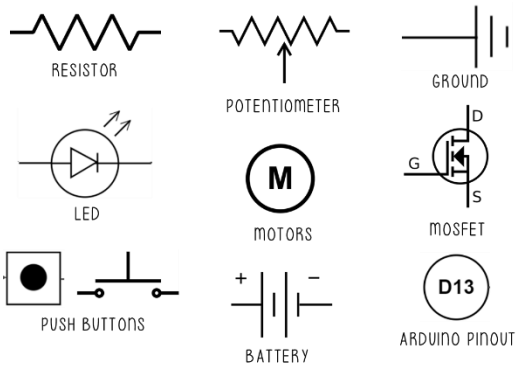


Note: Comments on the "completed code" section of the lessons are confusing. When they are later referenced, students think that they must include comments for the code to upload and function as expected.

The usefulness and purpose of commenting code within the IDE are explained in the microcontroller basics under Arduino integrated development environment in a section titled IDE coding setup and loop. Computer programs can be annotated by adding Human Readable Descriptions that explain what the code is doing. When comments are used correctly, they can greatly simplify code upkeep and speed up bug discovery.

Note: Explain circuit symbols. There doesn't seem to be any understanding of what they mean from the get-go.

Circuit symbols are a section of the microcontroller basics under circuitry basics that goes into great depth and explains the function and implementation of circuit symbols. The diagram below, which shows all circuitry symbols used in the lessons, is the revised section's main visual feature:

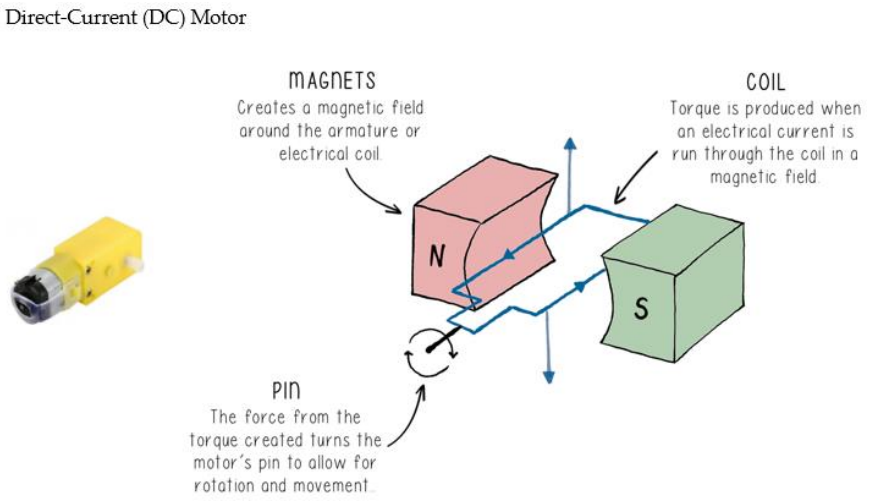


Note: Explain elements. Maybe a diagram of a potentiometer? Explain what is happening as voltage and resistance change in the potentiometer.

Note: Make a picture example of what a DC motor is and how it works – needs a diagram to explain how it works.

Note: Include a picture and diagram breakdown of a MOSFET

In the added section labeled "Component explanations" of the microcontroller basics chapter on circuitry, you can find a list of circuit elements, their function, implementation, and diagrams. Here is an illustration of the revised decomposition of a direct-current (DC) motor as a circuit component:



3.2.5 Project

Note: Discussed with students what they might change about the project to make it more intriguing or use the lessons more effectively within the project. Ideas from them included maybe having it set up as a competition and less information on the project page, so students have to figure it out on their own.

The final project was established to be a road course wherein students have to make a cart build throughout the lessons and by gathering data to determine the cart's velocity, guide it through a designed path created by the teacher (by using tape or something similar on the floor). The section itself has been revised to include less "guidance" and removed all step-by-step solutions allowing students to use critical thinking and their knowledge after going through the manual's lessons. An element of competition can also be added at the teacher's discretion.

3.3 Discussion

As previously discussed, educational standards have quickly changed from content-based lectures and memorization to hands-on learning that promotes creativity, problem-solving, and cognitive engagement. This lesson manual was designed as introductory material for microcontrollers to teach electrical circuitry to meet and offer ways to surpass secondary education standards and requirements concerning electrical circuitry, particularly in the state of Utah. By applying electrical principles physically, this capstone aimed to give students and aspiring instructors a fundamental understanding of microcontrollers.

Throughout the testing process, the undergraduate students expressed their admiration and excitement about using it in their future classes. Various comments were made and noted about how enjoyable the material became whilst learning, and how they learned to work with Arduinos and felt confident doing so in the future. Primarily, students remarked on how they could use them in their classrooms in a variety of other ways to teach a wide range of physics topics. With the help and input of the BYU students, the manual was revised to be optimal in encouraging individual curiosity whilst teaching the foundational knowledge of designing circuits, and software design made with simple objectives in mind such as input/output interactions of lights, motors, and sensors inspiring the next generation of STEM students (Litts et al. 2017).

Whilst teaching electricity, teachers must require students to define the problem, identify criteria and constraints, develop possible solutions using models, analyze data to make improvements from iteratively testing solutions and optimize a solution. The revised manual and its designed structure offer sections of precise direction in the hopes of laying down foundational understanding (mainly the microcontroller basics and lesson sections) followed by moments that encourage exploration and stimulate interest in understanding natural phenomena, such as the exploration sections within lessons and the final project. In response to Utah's education requirements, the manual strives to perform "guided research" in science education whilst allowing students to construct and reflect learned principles, both by themselves and in collaboration with others.

Learning is not merely an act of retaining information but creating ideas informed by evidence and linked to previous ideas and experiences whilst building an interest in

understanding natural phenomena, encouraging students to find applications of concepts in everyday life boosts the enjoyable, creative, and inquisitive nature of physics.

Conclusion

The capstone sought to offer students and future teachers a fundamental grasp of microcontrollers by physically implementing electrical principles, and based on feedback from undergraduates, it appears that the manual succeeded in its intended purpose. As teachers continue to find new ways to incorporate hands-on teaching and learning into their classes, it is hoped that mediums like these will become valuable assets in teaching physics concepts when encountering physical implementations. Future work can expand on the simple format of the presented manual to include more lessons and projects that broaden the teacher's and students' understanding of how microcontrollers can be a helpful tool for exploring real-world phenomena other than simple electricity, such as classical mechanics. I hope that my manual will be useful in secondary education classes, as well as for undergraduate students at BYU studying to become secondary education teachers, to learn how to work with and explore the world with microcontrollers, either as a formatted example for lesson planning or through direct application of the manual.

References

ATEŞ, Özlem, and Ali ERYILMAZ. “Effectiveness of Hands-on and Minds-on Activities on Students’ Achievement and Attitudes towards Physics.” *Asia-Pacific Forum on Science Learning and Teaching*, vol. 12, no. 1, ser. 6, June 2011.

Aycan, Ş. & Yumuşak, A. (2003). Lise fizik müfredatındaki konuların anlaşılma düzeyleri üzerine bir araştırma [A study on the levels of understanding of high school physics curriculum subjects], National Education Journal, 159, 171-180.

Aykutlu, I., Bezen, S., & Bayrak, C. (2015). Teacher opinions about the conceptual challenges experienced in teaching physics curriculum topics. Procedia - Social and Behavioral Sciences, vol. 174, pg. 390–405.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the National Academy of Sciences, 111(23), 8410–8415.

Hurca, N. (2012). The Influence of Hands-on Physics Experiments on Scientific Process Skills According to Prospective Teachers’ Experiences, European Journal of Physics Education, Vol 4, Issue 1, 2013.

Karakuyu, Y. (2008). Problems of Physics Teachers in Physics Education: Afyonkarahisar Sample, Mustafa Kemal University Journal of Social Sciences Institute, 10, 147–159.

Kontra, Carly, et al. “Physical Experience Enhances Science Learning.” *Psychological Science*, vol. 26, no. 6, 2015, pp. 737–749.

Litts, Breanne K., et al. “Stitching Codeable Circuits: High School Students' Learning about Circuitry and Coding with Electronic Textiles - Journal of Science Education and Technology.” *SpringerLink*, Springer Netherlands, 29 May 2017.

Loewenberg Ball, D., Thames, M. H., & Phelps, G. Content Knowledge for Teaching: What Makes It Special? Journal of Teacher Education, 59(5), 389–407, 2018.

Marzoli, Irene, et al. “Arduino: From Physics to Robotics.” *SpringerLink*, Springer International Publishing, 1 Jan. 1970.

Rillero, P. (1994). Doing science with your children. East Lansing, MI: National Center for Research on Teacher Learning (ERIC Document Reproduction Service No. ED 372 952).

Santosa, E. S. B., and S. Waluyanti. "Teaching Microcontrollers using Arduino Nano Based Quadcopter." Journal of Physics: Conference Series, vol. 1413, no. 1, 2019.

Slugan, Jelena, and Ivica Ružić. “High School Stem Curriculum and Example of Laboratory Work That Shows How Microcomputers Can Help in Understanding of Physical Concepts.” *World Academy of Science, Engineering and Technology International Journal of Educational and Pedagogical Sciences*, vol. 11, no. 8, 2018.

USBE- Utah State Board of Education. https://www.schools.utah.gov/File/e5d886e2-19c3-45a5-8364-5bcb48a63097.

Appendix A. Completed Lesson Manual

Microcontroller Basics Lesson Manual

Capstone 2022-2023

Caitlin Mastroianni

Table of Content

INTRODUCTION

PURPOSE

The discussion surrounding the benefits of "hands-on" experimentation in high school studies has been an area of great study, especially in the conceptualization of topics taught in physics classrooms. The foundation upon which the capstone is built is in response to two areas of study: the realized difficulty in a student's conceptualization of electricity and the benefits of hands-on learning in understanding physical concepts.

Studies both in 2002 and 2008 determined that students struggling to understand concepts explored and taught in secondary-level physics could be categorized into three main groups: "students lacking background about the physics subjects (before high school)", "students who were not familiar with the subject from daily life" and "students who couldn't embody abstract concepts". (Aycan & Yumuşak, 2002; Karakuyu, 2008). Teachers echoed these results in various studies published in the field of Educational Sciences, stating that students gravely struggle to understand variable currents, capacitors, coils, transformers, and electronic circuit elements in electricity units taught in secondary education. "It was concluded that the challenges … stemmed [from] various factors such as students' lack of knowledge in terms of concepts, the existence of misconceptions, difficulties in comprehending abstract concepts, deficiencies in mathematical operations, and insufficiency of time allocated to the course leading to incomplete teaching tasks." (Aykutlu, I., Bezen, S., & Bayrak, C. 2015). Appealing to the weakness in the educational structure, and the lack of foundational knowledge with which students enter secondary-level physics, this capstone is constructed with students in establishing connections between electricity and circuitry through hands-on means.

Hands-on experimentation has been proven to be extremely beneficial to understanding and conceptualizing physical structures. An intriguing study, published in 2015, explored the realm of hands-on learning in testing whether physical experience with angular momentum "increases the involvement of sensorimotor brain systems during students' subsequent reasoning and whether this involvement aids their understanding." The results produced directly correlated the benefits of such physical interaction, stating that it "leads the way for

classroom practices in which experience with the physical world is an integral part of learning." (Kontra, C., Lyons, D. J., Fischer, S. M., & Beilock, S. L. 2015). With real-world applications, students can become confident to explore barriers and expand their learning through trial and error.

This capstone aims to provide a framed structure through a tutorial manual in which both teachers and students can learn and apply principles of electricity and circuitry in a physical space. It becomes apparent that physics instructors should prepare syllabuses that build from low-cost materials and experiences in which students can enhance their scientific process skills through personal discovery and experimentation (Hurca .N. 2012).

WHAT ARE MICROCONTROLLERS?

A microcontroller unit, or MCU for short, is a single integrated circuit comprised of various elements depending on the purpose or brand. Elements found on MCUs typically include a microprocessor, input/output ports (I/O), random access memory (RAM), read-only memory (ROM), timers, and counters accessed by varying ports and pins on the unit itself.

Arduino is an open-source electronics platform based on easy-to-use hardware and software.[1] The simplistic design of the single-board microcontroller, along with microcontroller kits, allows for the easy building of digital devices and is widely available, and a low-cost option.

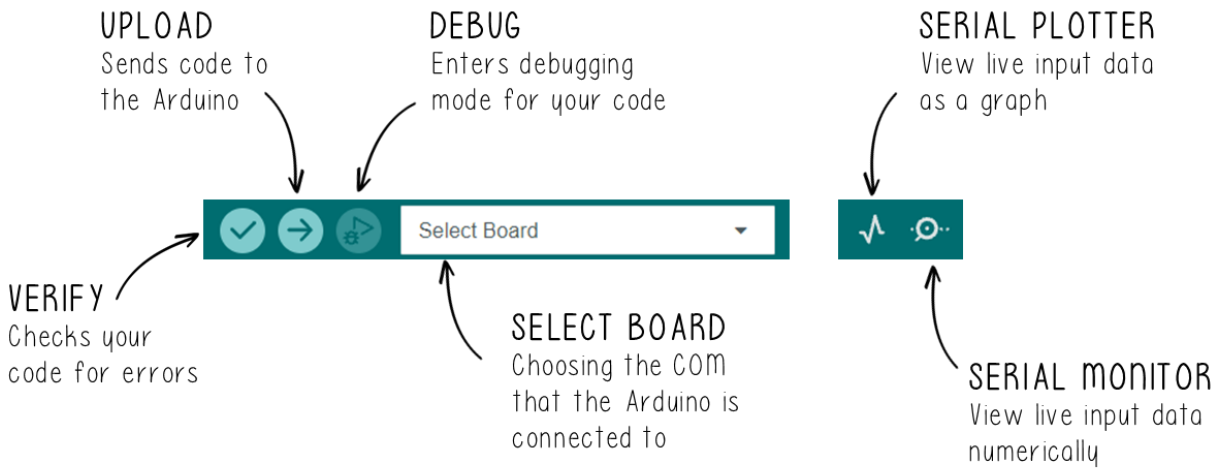
2015). Teacher opinions about the conceptual challenges experienced in teaching physics curriculum topics. Procedia - Social and Behavioral Sciences, 174, 390–405.
Aycan, Ş. & Yumuşak, A. (2003). Lise fizik müfredatındaki konuların anlaşılma düzeyleri üzerine bir araştırma [A study on the levels of understanding of high school physics curriculum subjects], National Education Journal, 159, 171-180.
Hurca, N. (2012). The Influence of Hands-on Physics Experiments on Scientific Process Skills According to Prospective Teachers' Experiences, European Journal of Physics Education, Vol 4, Issue 1, 2013.
Karakuyu, Y. (2008). Problems of Physics Teachers in Physics Education: Afyonkarahisar Sample, Mustafa Kemal University Journal of Social Sciences Institute, 10, 147–159
Kontra, C., Lyons, D. J., Fischer, S. M., & Beilock, S. L. (2015). Physical Experience Enhances Science Learning. Psychological Science (0956-7976), 26(6), 737–749.

MICROCONTROLLER BASICS

ARDUINO INTEGRATED DEVELOPMENT ENVIRONMENT

To download the Arduino IDE for free, head to https://www.arduino.cc/en/software

IDE Button Functions



Verify – checks your code for errors whilst compiling it; if there are errors, they will appear in the interface below with a line number and an error description.

Upload – compiles the code and uploads it to the specified board

Debugger - runs the code piece by piece, monitoring everything that happens as the program runs. This also allows you to pause the app at any point to examine its state then step through your code line by line to watch every detail as it happens by setting a breakpoint (this is done by clicking on the left margin of the code, just to the left of the code line number, and is represented by a red dot).

Select Board – selection the type of board that is plugged in

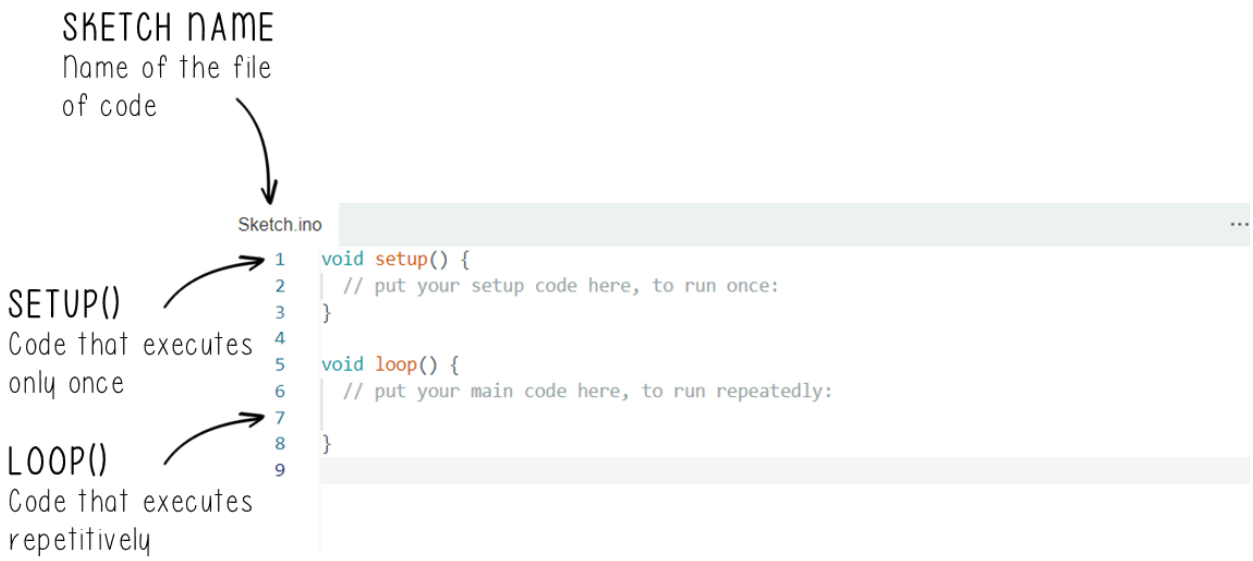
How to select a board

Select your COM based on the external port that you are using on your laptop/desktop. If you are unsure of the exact port's name or number, plug in your Arduino and see which port becomes available and becomes unavailable once you unplug your Arduino.

Serial Plotter – whilst an Arduino is plugged in, data being collected by the board can be analyzed visually through graphing in real-time

Serial Monitor – whilst an Arduino is plugged in, data being collected by the board can be analyzed visually through terminal data in real-time

IDE Coding Setup and Loop



Setup – the sketch's starting point. This is where variables and pin modes should be initialized as well as accessing libraries, etc. The setup function will only run once and will re-initialize each time the Arduino is powered up/reset.

Loop – the sketch's looping point. This is where the actions are coded in the potions loop consecutively and will continuously perform the codes without resetting the collected data. The Arduino is actively controlled by the looping portions of its code.

Output Window

Output – communication between the computer and the Arduino. This window will declare success or may display any errors present during compiling and/or uploading.

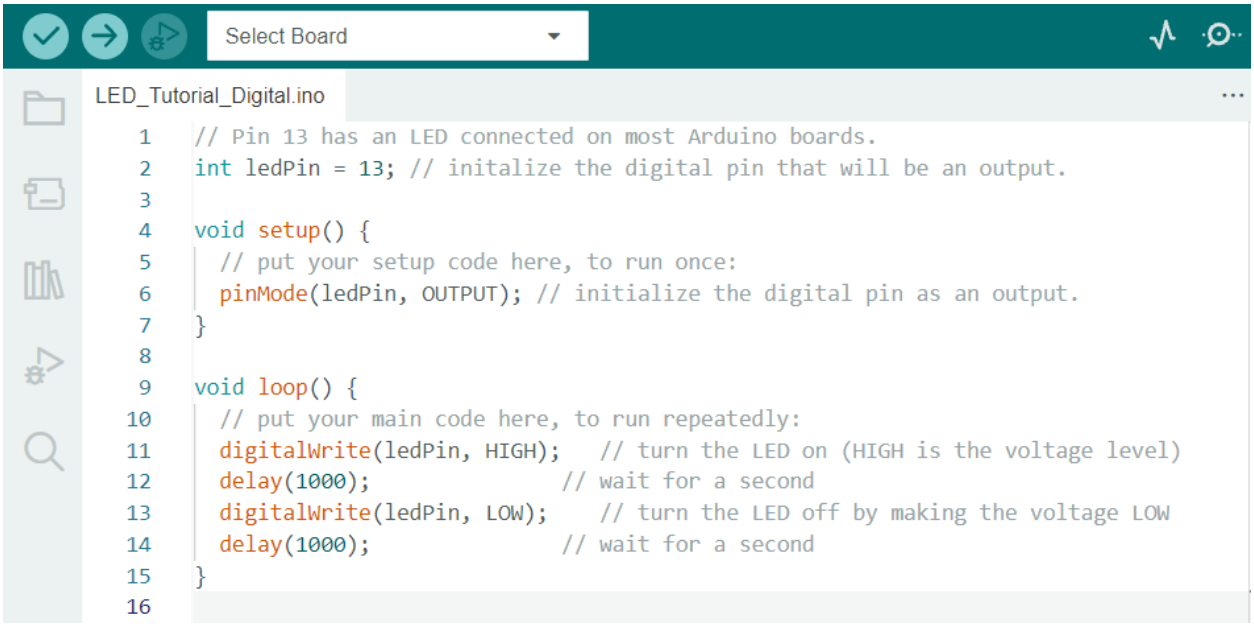
ARDUINO IDE CODING ESSENTIALS

Description of the coding essentials for the Arduino IDE including the most frequent functions, and coding examples with how to implement common functions.

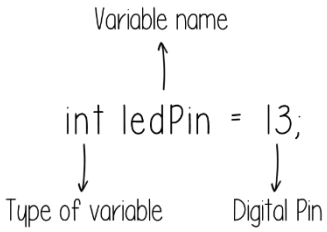
Most Frequently Used Functions

Basic Function	**Description**
digitalRead()	Reads the value from a specified digital pin, either HIGH or LOW.
digitalWrite()	Write a HIGH or LOW value to a digital pin.
pinMode()	Configures the specified pin to behave either as an input or an output.
analogRead()	Reads the value from the specified analog pin.
analogWrite()	Writes an analog value (PWM wave) to a pin.
delay()	Pauses the program for time (in milliseconds) specified as a parameter.
Serial.begin()	Sets the data rate in bits per second (baud) for serial data transmission (default set to 9600 bits per second).
//Enter comment here	Commenting ability on code.

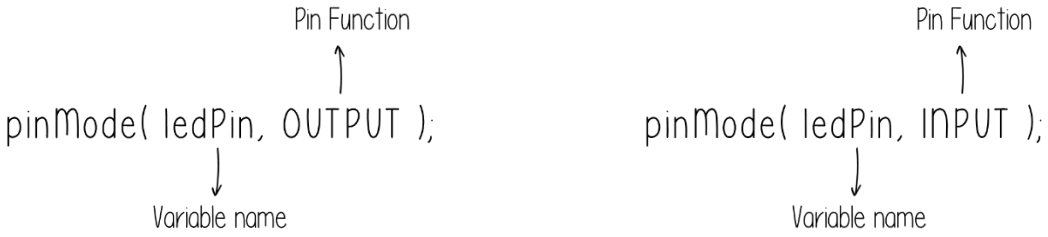
Example Implementations of Common Functions (DIGITAL)



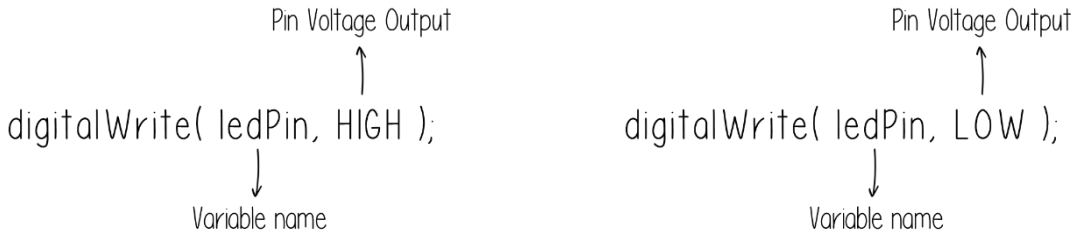
Initialization, Line 2 - Initializing variables with names. Variables are initialized outside of the setup loop, with the pin's functions initialized within the setup loop. In this example, *ledPin* is the name of the object that is connected to the digital pin of integer 13. The format for initializing variables is as follows:



pinMode(), Line 6 – Initializing a pin's function. A variable's function that has been previously named through an initialization (e.g., line 2) is defined by using pinMode(); this configures the specified pin to behave as either an INPUT or an OUTPUT. The format for initializing a pin's mode is as follows:



digitalWrite(), Line 11/13 – Controlling a pin's output. A digital pin can be controlled by calling digitalWrite() and writing the digital pin to a HIGH or LOW state. When the pin is initialized as an output, its voltage will be set to 5V when at HIGH and 0V when at LOW. The format for writing a value to a digital pin is as follows:

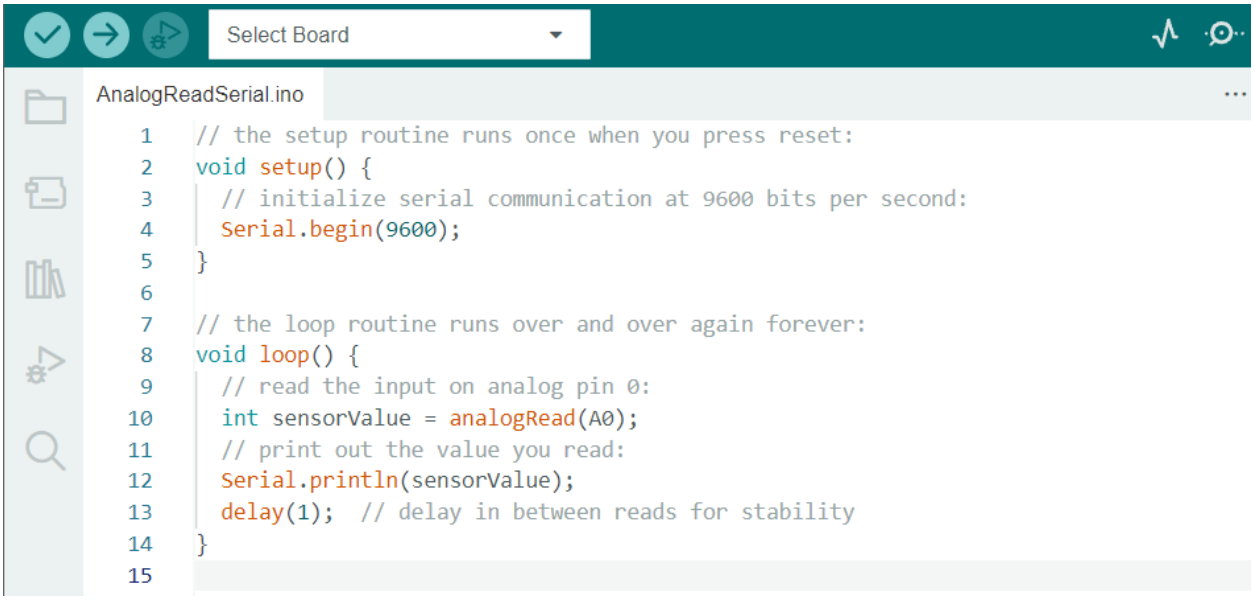


delay(), Line 12/14 – Setting a line's run length. Once an action is called such as digitalWrite where a pin is set to HIGH, delay can be used to separate the previous line of code's call to the next by a set amount of time. This will pause the program for the time (in milliseconds) specified as a parameter. The format for writing a delay is as follows:

```
delay( 1000 );
```

Delay time in milliseconds

Example Implementations of Frequent Functions (ANALOG)

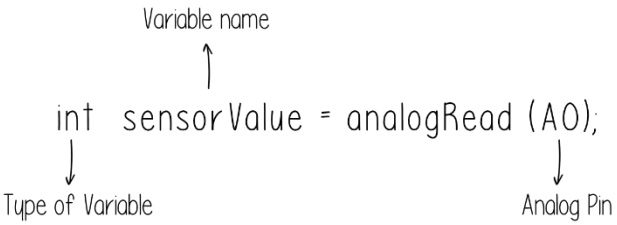


Serial. begin(), Line 4 – Initializing the serial communication. By initializing the serial communication, the user can access the input data from the Arduino board in real-time using the serial monitor or serial plotter in the Arduino IDE. The format to initialize the serial communicator is as follows:

```
Serial.begin( 9600 );
```

Communication rate in bits per second

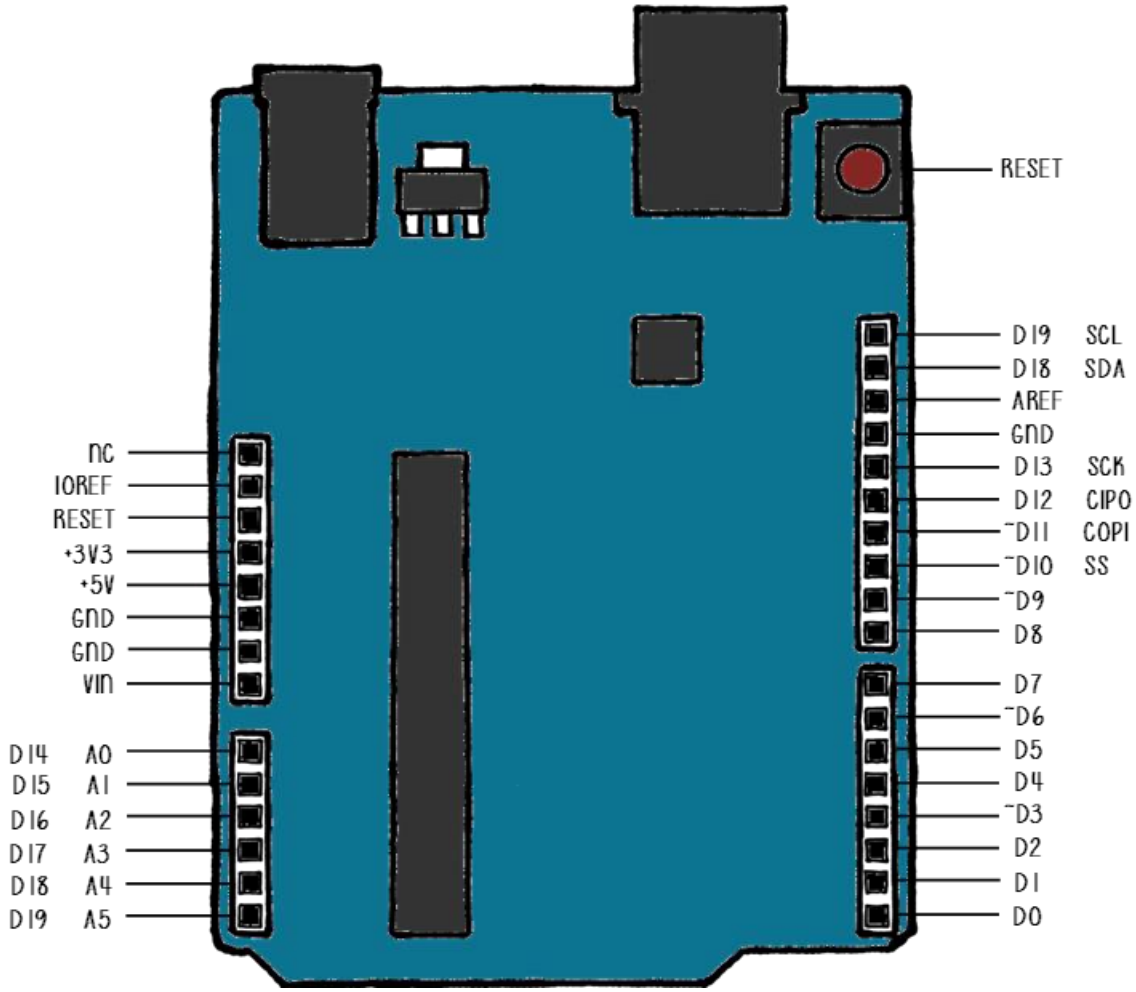
analogRead(), Line 10 – Initializing the collected values of data from the analog pin. At the beginning of the continuous loop and in each continuous run-through, a value is set from the data collected when analogRead() is called (analog returns values between 0 and 1023 through a 10-bit analog to digital converter). That value read is then printed in the serial communicator's window. The format for accessing the analog pin's input is as follows:



Serial.println(), Line 12 – Outputting collected data to the serial communicator. The print line call accesses the input value initialized through analogRead() and outputs it into the serial communicator accessed through the serial monitor or serial plotter. The format for printing the analog pins input to the serial communicator is as follows:

digitalRead() – Accessing the collected data from an INPUT pin. Reading a digital pin allows the user to see the information collected from the Arduino in real time through the serial monitor (digital returns HIGH or LOW). The format for accessing the digital pin's input is as follows:

ARDUINO SCHEMATICS



Power Supply - There are three ways to power your Arduino depending on what power outage you need and if the Arduino needs to be functioning freely or can be powered by a chord.

USB Plug - The most common way to power an Arduino board is by using its onboard USB/micro-USB connector. The USB connector provides a regulated 5V line to power the board's electronics.

Power Jack & External Power Supply - The recommended voltage and current ratings for external regulated DC power supplies connected to the barrel jack connector are as follows: voltage (7-12 V) and current (1 A). Exceeding recommended voltage, the regulators might overheat whilst not reaching a 7-volt input might not suffice the powering of the board along with connected components.

Ground Pins (GND) - All GND pics are used to close an electrical circuit and are interconnected with one another. There are 5 found within the board.

Reset – pin resets the Arduino

IOREF - This pin is the input/output reference. It provides the voltage reference with which the microcontroller operates.

Digital I/O Pins – These digital pins can be used as both input and output pins, defaulting to inputs if not explicitly declared as outputs in the Arduino software. Digital pins represent voltage in 1 bit: represented as either 1 or 0, with no in-between.

Digital OUTPUT - When configured as outputs, these pins are capable of being either on or off, these pins either output on a HIGH voltage state of 5V and when OFF in a LOW voltage state of 0V.

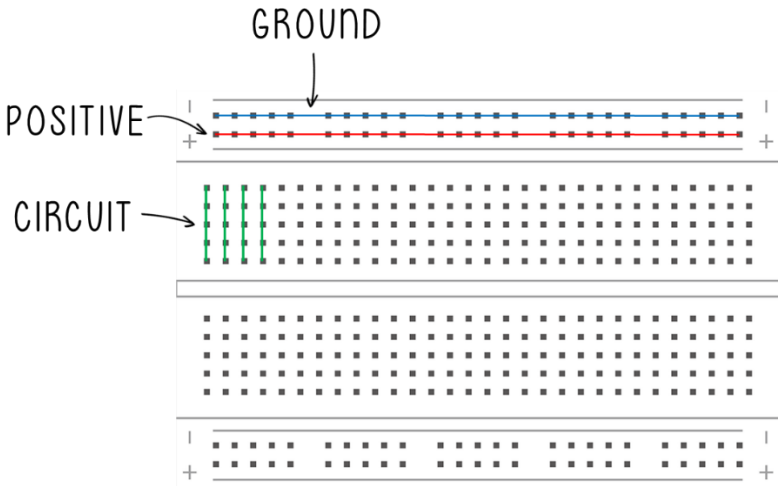
Digital INPUT - When in their default state, the voltage being read is converted into 1 bit whereas any voltage being read below 0.8 V is read as 0 and once passing this threshold convert into a 1.

Analog Pins – These analog pins use ADC (analog to digital converter) to serve as analog input but can also function as digital inputs or digital outputs.

Analog INPUT - Capable of reading analog voltages on a 10-bit resolution meaning that it can represent analog voltage by 1,024 digital levels.

CIRCUITRY BASICS

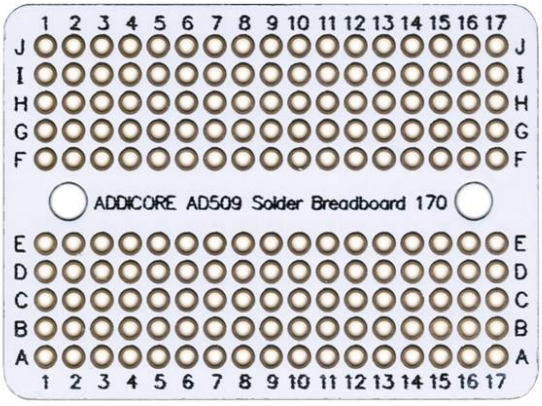
Breadboard Diagram



A breadboard is a plastic board with interconnected pins that are used to connect electrical components to create a functional circuit. There are two kinds of strips – bus or power strips and terminal strips. Terminal strips (green) are used to connect the electrical components, each strip is connected vertically but not horizontally. Bus or power strips (red and blue) are used to connect power or ground to the terminal strips, these strips are connected horizontally to one another.

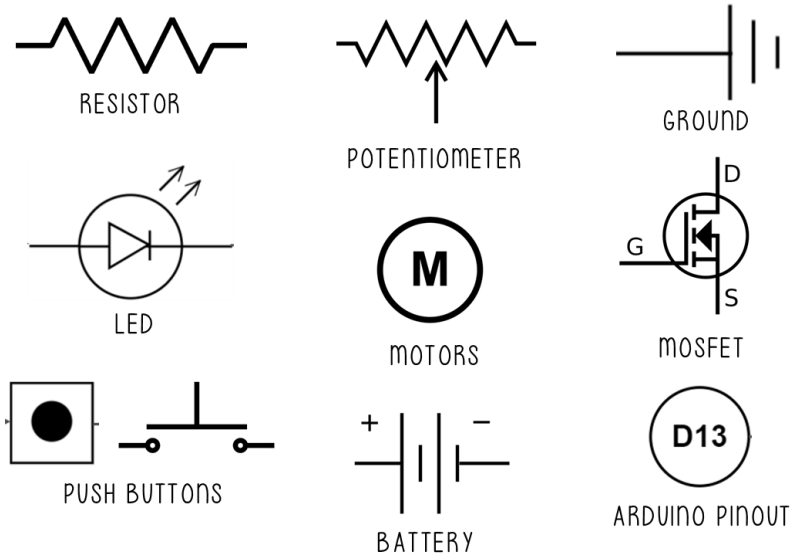
Breadboards are used to build non-permanent circuitry. These come in handy when testing, creating, and experimenting with circuits, allowing for easy subtraction, addition, and analysis of various electrical components.

For more permanent circuitry, soldering breadboards can be used to solder electrical components onto it – like the one displayed below:



Circuit Symbols

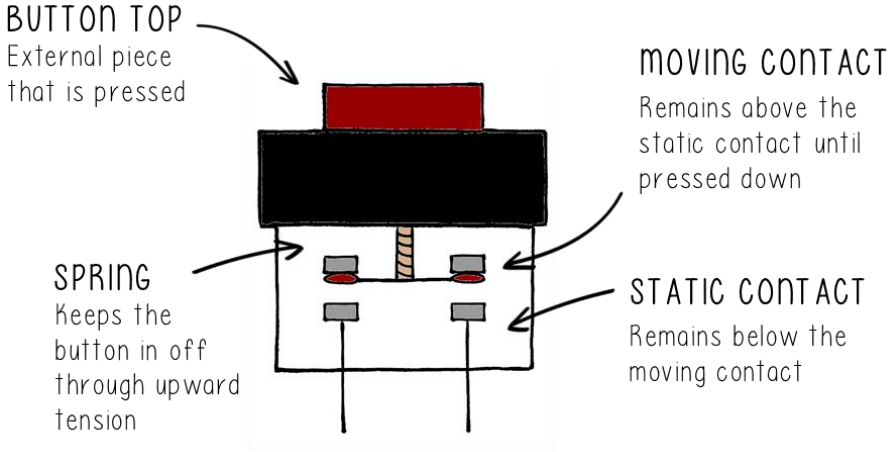
Below are the circuit symbols that are both common in simple circuitry and are used in the lessons within this workbook.



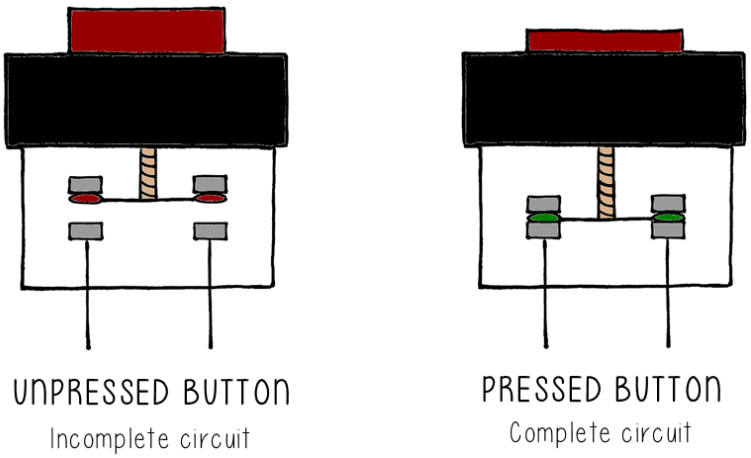
All electrical circuitry is described through universal electrical symbols – they are a graphical representation of circuits. Think of it as a universal language. All circuits start with a symbol and end with a symbol and all symbols in between are connected with plain lines.

Component Explanations

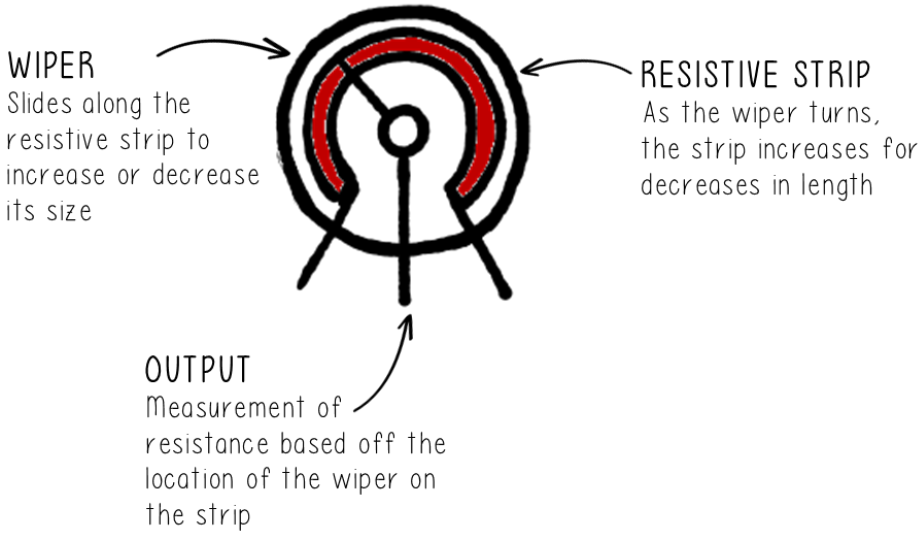
Button



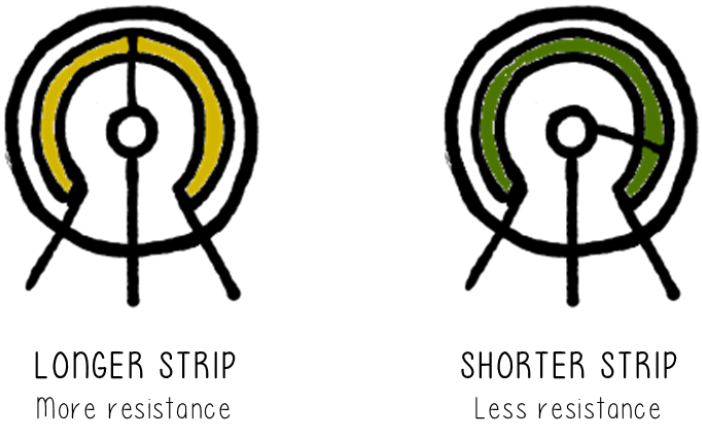
Until pressure is applied on the button's top, the moving contact will remain separated from the static contact which prevents the circuit from being completed. Once the button is pressed, the contacts connect, and the circuit is completed allowing the voltage to flow freely.



Potentiometer

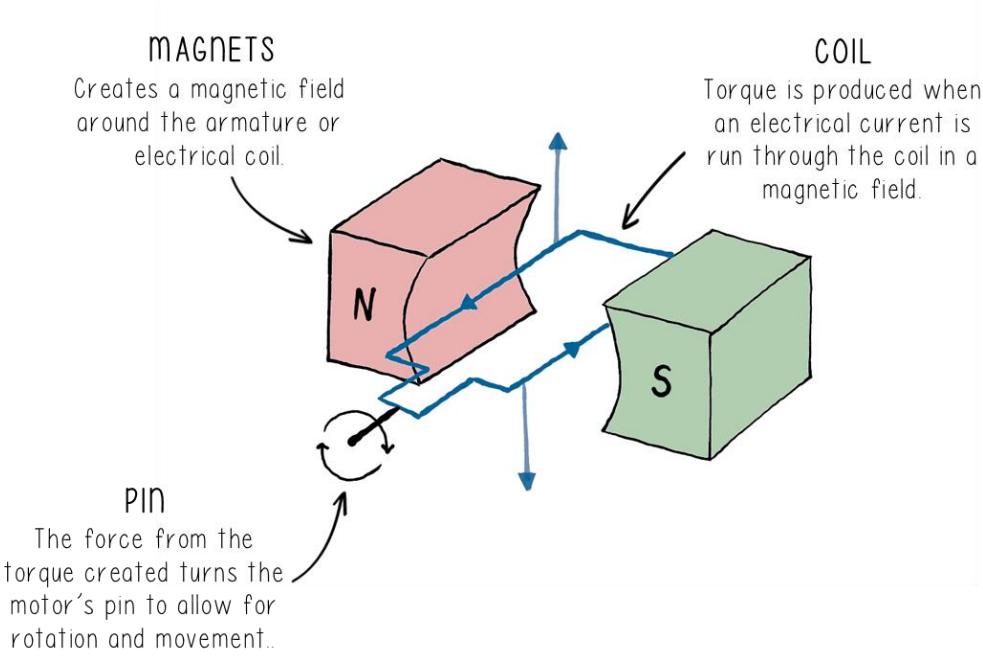


The input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the commencing point and the point where the sliding contact lays.



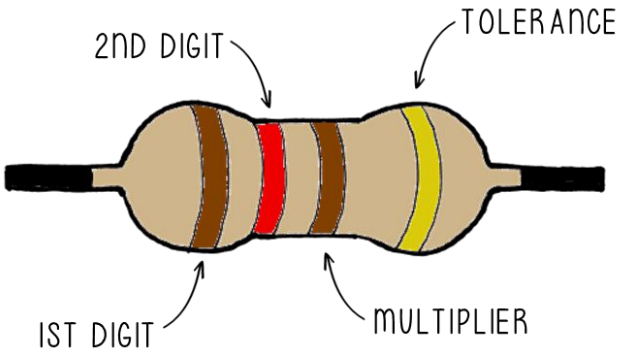
As the dial on the potentiometer is turned, the position of a sliding contact wiper across the resistance strip changes and therefore, increases or decreases the total resistance.

Direct-Current (DC) Motor



DC motors take electrical power through a DC current and convert the energy into torque and mechanical rotation within the motor. Within the motor, magnetic fields are created to allow an electrical coil to rotate when an electrical current is running through it. When the two interact, the torque created generated the mechanical rotation of a pin.

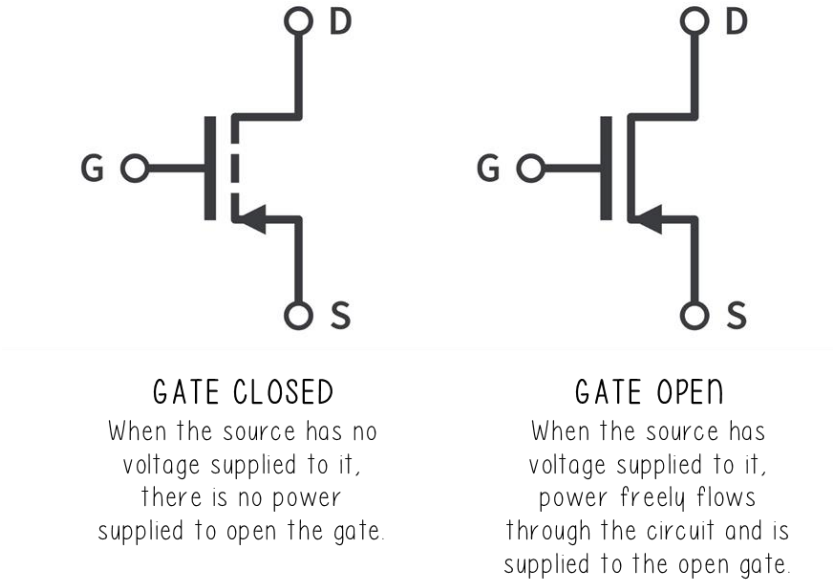
Resistor



A resistor is a passive two-terminal component that limits the current flowing through itself by creating resistance through heat. The resistance of the resistor is determined by a ceramic rod wrapped with a copper wire – the thinner the copper wire, the higher the resistance, and the same stands for a thicker wire having a lower resistance. The colored bands on a resistor demonstrate the resistance of the resistor. The table below explains each color's meaning and representation:

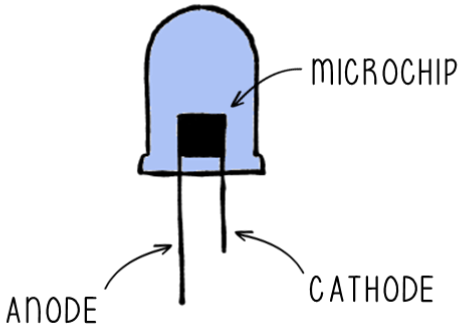
Color	1st, 2nd Band Significant Figures	Multiplier	Tolerance
Black	0	x 1	
Brown	1	x 10	± 1% (F)
Red	2	x 100	± 2% (G)
Orange	3	x 1K	± 0.05% (W)
Yellow	4	x 10K	± 0.02% (P)
Green	5	x 100K	± 0.5% (D)
Blue	6	x 1M	± 0.25% (C)
Violet	7	x 10M	± 0.1% (B)
Grey	8	x 100M	± 0.01% (L)
White	9	x 1G	
Gold		x 0.1	± 5% (J)
Silver		x 0.01	± 10% (K)

Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET)



A MOSFET (metal-oxide-semiconductor field-effect transistor) has three pins: a source, a gate, and a drain. The MOSFET works in the same principle as a button and a switch – the MOSFET controls the voltage and current flowing within the circuit from the source to the drain.

Light-Emitting Diode (LED)



As an electrical current passes through the microchip, it illuminates and creates the visible light that we identify as a light-emitting diode (LED). The long pin, called an anode, is connected to power or a voltage supply whilst the short pin, called a cathode, is connected to negative or ground. Current in an LED flows from the anode to the cathode and never in the opposite direction.

LESSONS

OVERVIEW

Lesson Kits

For each of the following lessons, be sure to have the following listed below:

Required:

- Arduino Uno
- Arduino IDE
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- LED (Standard 2-Pin, Single Color) x 1
- 220-ohm Resistors x 2
- Potentiometer x 1
- MOSFET x 1
- Car Kit that includes DC Motor (Standard 12VDC) x 4

Optional:

- LED (Standard 4-Pin, RGB) x 1

Lesson Structure

Each lesson will be split into parts allowing those working through it to understand each task completely whilst also encouraging personal exploration of circuitry and coding. The lessons are structured in parts as follows:

- Hardware Required
- Schematic
- Step-By-Step
- Completed Code
- Exploration Questions

Blink 1 - LEDs

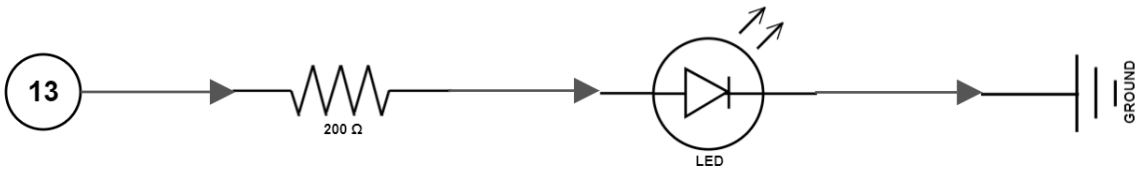
Hardware Required

- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- LED (Standard 2-Pin, Single Color) x 1
- 220-ohmhm resistor x 1

Schematic

An explanation of how LEDs work can be found on page

Connect the digital pin (pin D13) to one end of the 220-ohm resistor, followed by connecting the anode (the positive leg or longer leg) of the LED to the opposite side of the resistor. Connect the LED's cathode (the negative leg or shorter leg) to GND. Once your circuit is connected, move on to programming.



Step-By-Step Programming

Plug in your board to the computer and start the Arduino IDE.

Initialize digital pin 13 as the pin that your LED will be connected to. Digital Pin 13 is also connected to the LED onboard the Arduino Uno, allowing for easier circuit debugging, if necessary.

```
int ledPin = 13;
```

In your setup, initialize digital pin 13 to be an output pin by calling pinMode().

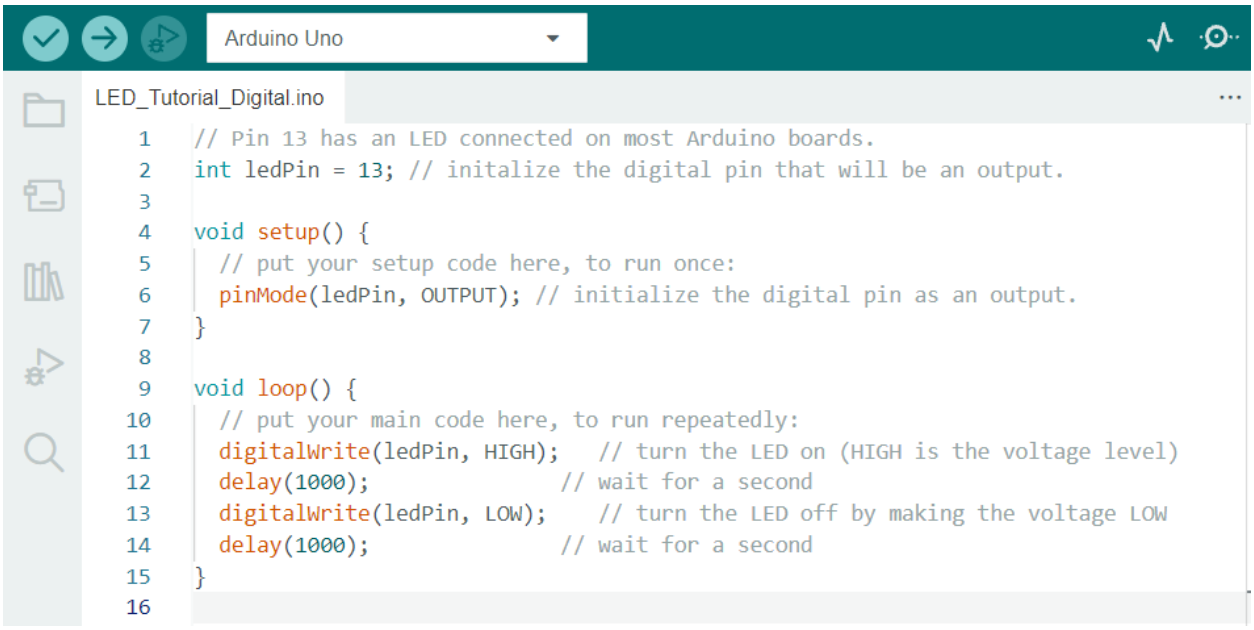
```
void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);
}
```

In your loop, make the LEDs blink; this means that a voltage will be provided so that the LED turns on for a specified period followed by removing the voltage provided so that the LED turns off. To do so, use digitalWrite() to declare whether the voltage is at HIGH or LOW followed by a delay().

```
void loop() {
 // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Compile your code to double-check that there are no errors and that the program itself will upload and run onto your chosen device. If the compilation is successful, upload the program and watch how the LED reacts.

Completed Code with Comments



Exploration

Delay()

Let's play around with the delay() function.

If we decrease a delay (e.g., line 12) what would the consequence be?

_____

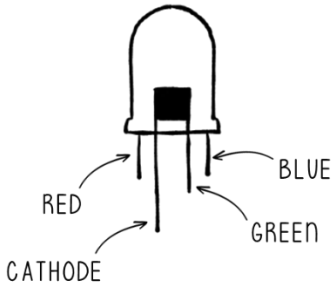
Turn down the delays on lines 12 and 14 to 500 and see how the LED reacts.

```
delay(500);
```

Taking what we now know about the delay function, configure the light to turn on for 3 seconds before turning off for 1 second.

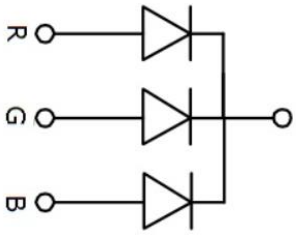
RGB Light-Emitting Diodes

RGB LEDs work just the same as a standard two-pin LED except that three pins are connected to different colored microchips to emit red, green, or blue light.



Let's connect the RGB to our Arduino so that we can separately control each of the colors emitted by the LED. We'll start with the circuit itself.
Draw below, using our RGB LED symbol, a circuit to make on your breadboard.

RGB LED Symbol: Circuit:



Taking your drawn circuit, let's add some code to our existing code from this lesson. Assign 2 new digital pins for your red and green LED pins.

```
int ledRedPin = (insert digital pin number here);
int ledGreenPin = (insert digital pin number here);
```

Now assign your variables to function as output digital pins in your setup() loop.

```
pinMode(ledRedPin, OUTPUT);
pinMode(ledGreenPin, OUTPUT);
```

In your loop(), turn on the digital pin assigned to your red LED pin to light up for 2 seconds then turn it off whilst the others take turns lighting up.

```
digitalWrite(ledRedPin, HIGH);
delay(2000);
digitalWrite(ledRedPin, LOW);
```

Repeat this process with the other pins. Your LED should now rotate between red, green, and blue light with each lasting for 2 seconds.

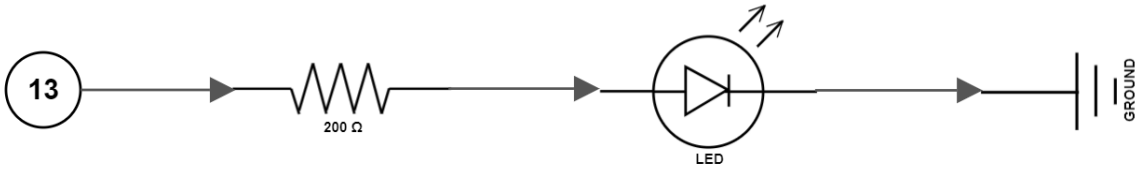
Blink 2 – LEDs and Potentiometers

Hardware Required

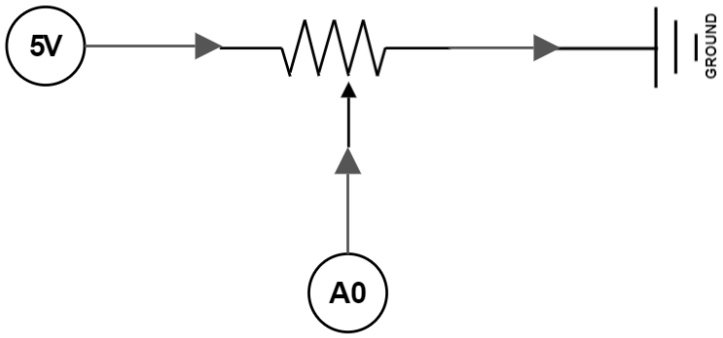
- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- LED (Standard 2-Pin, Single Color) x 1
- 220-ohmhm resistor x 1
- Potentiometer x 1

Schematic

Circuit One:
Connect the digital pin (pin 13) to one end of the 220-ohm resistor, followed by connecting the anode (the positive leg or longer leg) of the LED to the opposite side of the resistor. Connect the LED's cathode (the negative leg or shorter leg) to GND.



Circuit Two:
Connect the rightmost pin of the potentiometer to the positive, and the leftmost pin to ground. Connect the analog pin (pin A0) to the center pin.



Step-By-Step Programming

Continue to build from the previous code that you established in Blink 1.

Initialize two more variables that will be connected to the potentiometer – one variable will be to initialize which analog pin will be the input (e.g., A0); the other variable is to access that value that is collected at the moment a line that calls analogRead() is executed.

```
int ptPin = 0;
int ptValue = 0;
```

At the very beginning of the loop, set our access variable equal to the value being collected in our input port by calling analongRead().

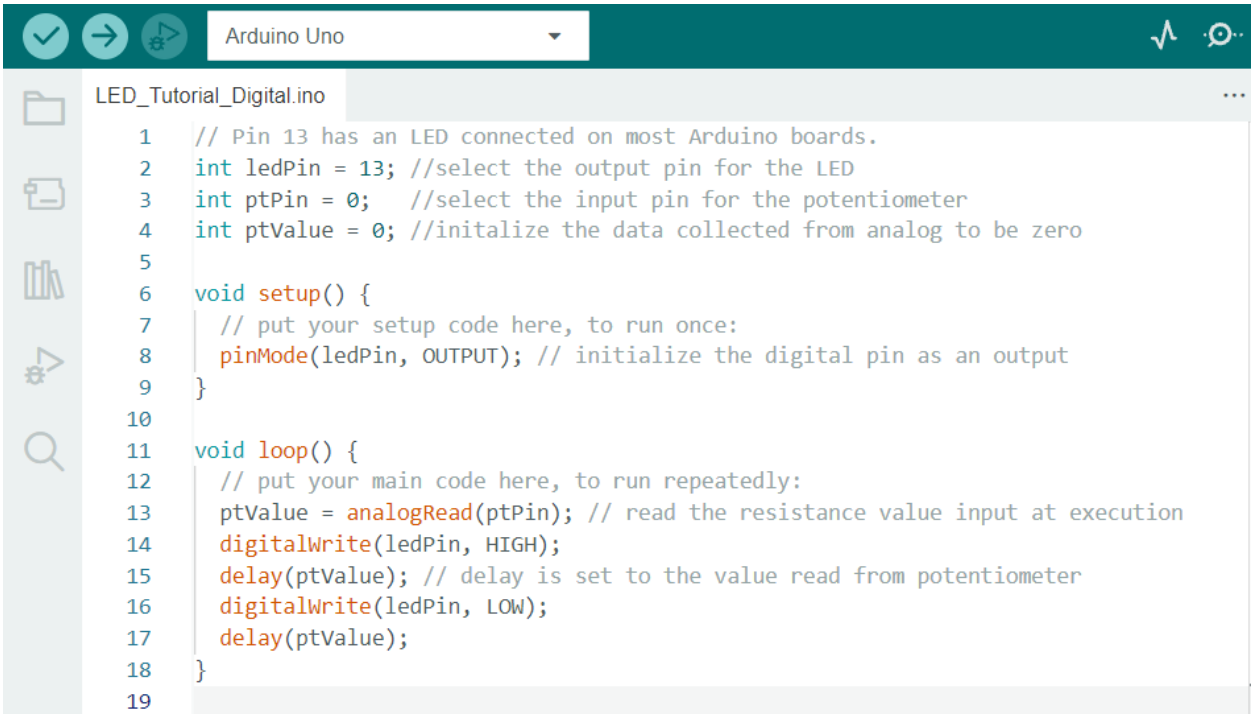
```
ptValue = analogRead(ptPin);
```

Set your delay() functions to the value collected from the potentiometer.

```
delay(ptValue);
```

Compile your code to double-check that there are no errors and that the program itself will upload and run onto your chosen device. If the compilation is successful, upload the program and watch how the LED reacts as you turn the potentiometer.

Completed Code with Comments



Exploration

Serial Monitor and Plotter

The serial monitor and plotter are great tools to see how the voltage input from an analog pin change over time with variables such as a potentiometer. Let's add two lines of code to see how the voltage changes as you spin the potentiometer's dial.

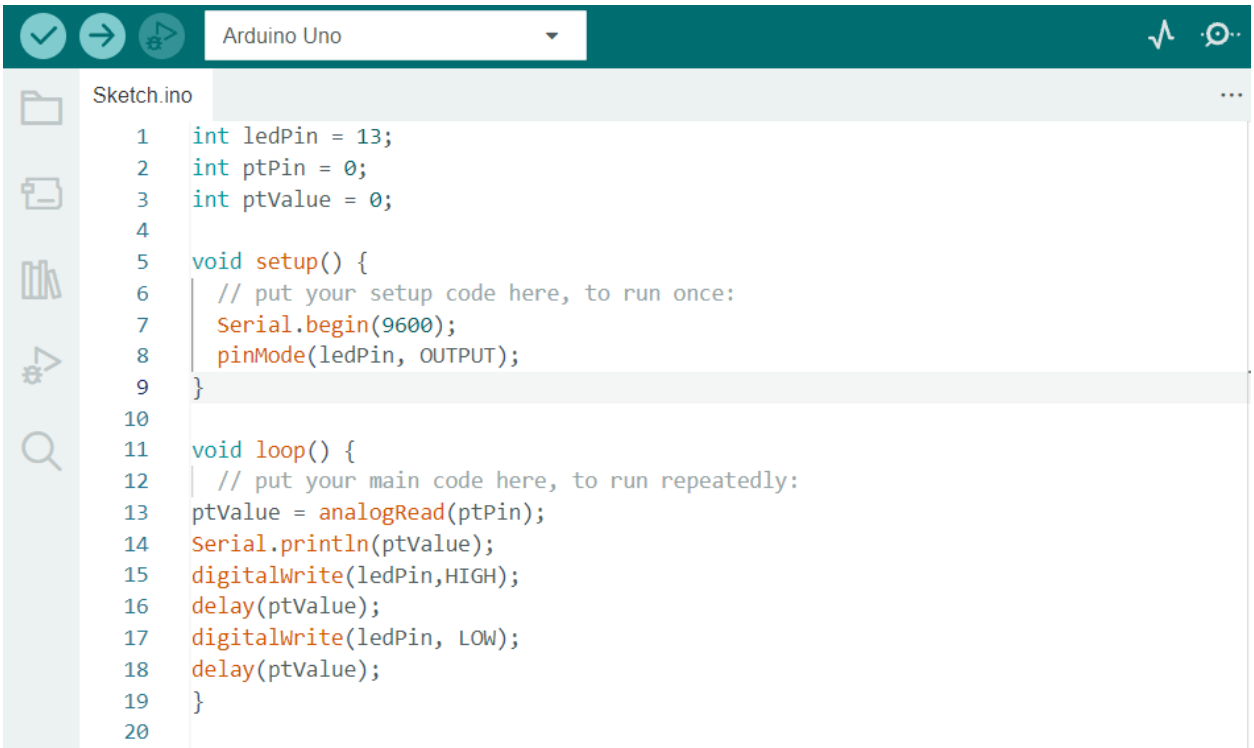
In our Setup() function, add the Serial.begin() command that declares a communication rate of 9500.

```
Serial.begin(9600);
```

In our Loop() function, add the Serial.println() command to print the values collected from our potentiometer after declaring the ptValue variable.

```
Serial.println(ptValue);
```

Your code should now look like this:



Upload your code to the Arduino and open your serial plotter on the top right of the Arduino IDE.

What behavior are you seeing in the serial plotter as you turn the dial and why?

_____

_____

_____

_____

Analog Output

Let's play around with the analog pins to see how they differ from digital pins.

When using analogRead, we used the value collected at the moment of execution to break up the changing of the output pin connected to the LED from HIGH to LOW. Why are digital pins so useful for making an LED blink?

_____

_____

_____

_____

Let's say that instead of making the LED go from on to off, we wanted to vary the brightness by changing the voltage supplied. Why would analog pins be better for performing this function?

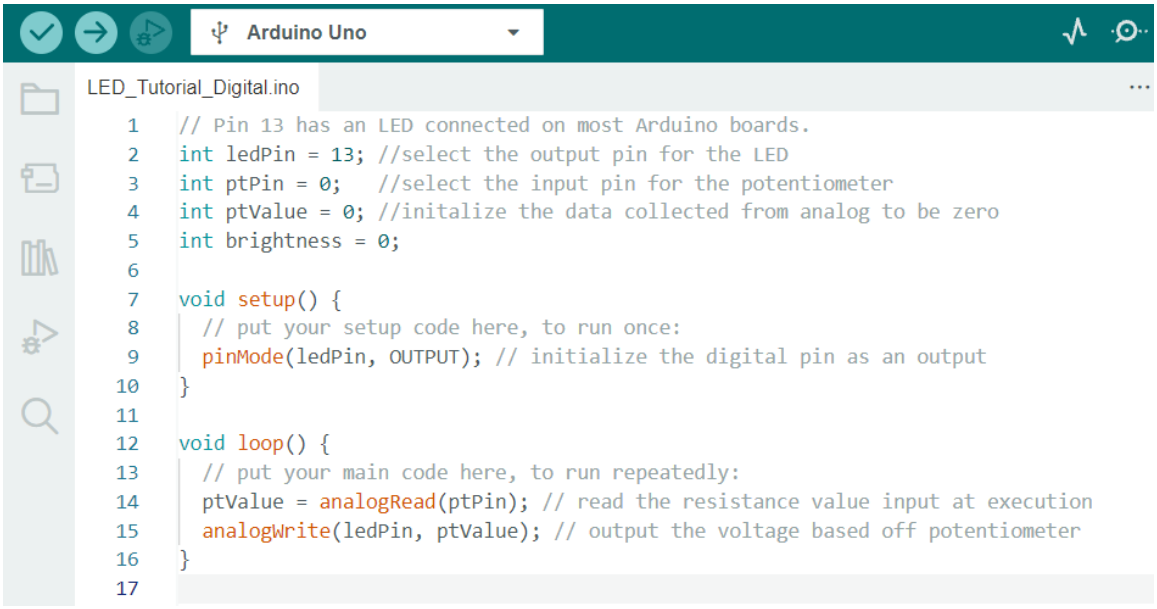
_____

_____

Change your code slightly to allow your potentiometer to decide the brightness of the LED. As we want our output to be on an analog scale, allowing us to access between 5V and 0V, replace our digitalWrite() in our loop function with analogWrite().

```
analogWrite(ledPin, ptValue);
```

Your code should now look like this:



Carefully (ever so slightly) twist your potentiometer to see how the LED reacts. As we slowly turn the potentiometer, the LED can be seen to have its brightness determined by the analog input by the analog output.

Blink 3 – LEDs and Buttons

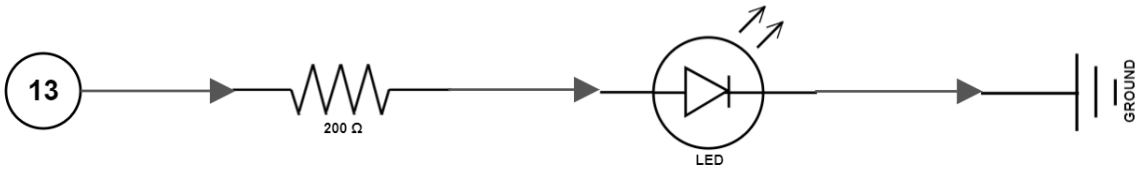
Hardware Required

- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- LED (Standard 2-Pin, Single Color) x 1
- 220-ohmhm resistor x 1
- Button x 1

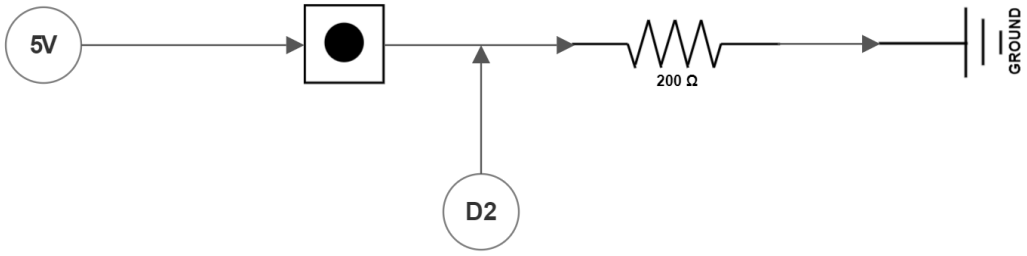
Schematic

The following circuits should be on two separate circuits.

Circuit One:
Connect the digital pin (pin 13) to one end of the 220-ohm resistor, followed by connecting the anode (the positive leg or longer leg) of the LED to the opposite side of the resistor. Connect the LED's cathode (the negative leg or shorter leg) to GND.



Circuit Two:
Connect the rightmost pin of the button to the voltage (positive), and the leftmost pin to the digital pin (D2) and a resistor. The other end of the resistor should be connected to ground.



Step-By-Step Programming

Continue to build from the previous code that you established in Blink 2's exploration.

Initialize two new variables that will be connected to the button – one variable will be to initialize which digital pin will be the input (e.g., D1); the other variable is to access what state the button is in and that will be collected at the moment a line that calls digitalRead() is executed.

```
int buttonPin = 1;
int buttonState = 0;
```

At the very beginning of the loop, set our access variable equal to the value being collected in our input port by calling digitalRead().

```
buttonState = digitalRead(buttonPin);
```

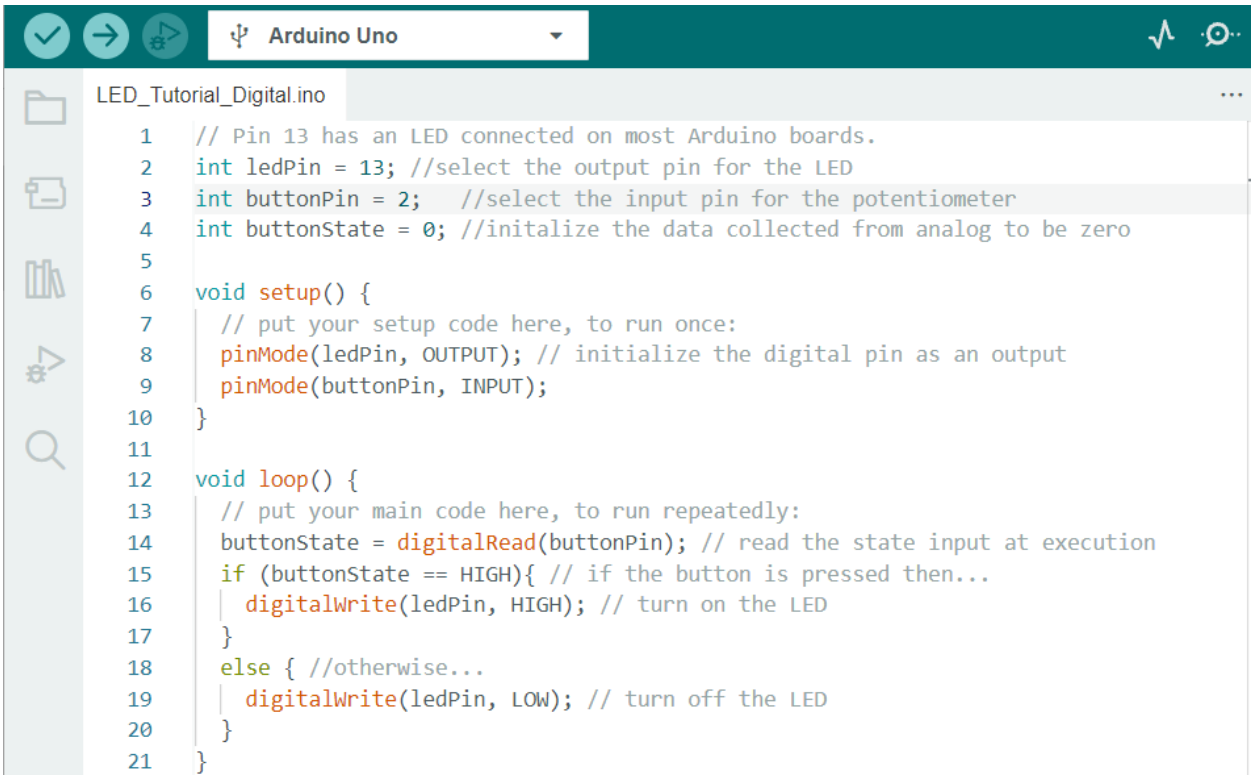
We will now implement an if-loop in which we declare what would happen depending on the specific circumstance – in this case, we will be looking into the state of the button to either turn on or off the LED. If our button is pressed down, the voltage that will be read will be in a HIGH state which will be programmed to turn on the LED. The opposite holds in the else portion of the if-else statement.

```
if (buttonState == HIGH){
  digitalWrite(LED, HIGH);
}

else {
  digitalWrite(LED, LOW);
}
```

Compile your code to double-check that there are no errors and that the program itself will upload and run onto your chosen device. If the compilation is successful, upload the program and watch how the LED reacts as you press the button.

Completed Code with Comments




```
// Pin 13 has an LED connected on most Arduino boards.
int ledPin = 13; //select the output pin for the LED
int buttonPin = 2;   //select the input pin for the potentiometer
int buttonState = 0; //initalize the data collected from analog to be zero

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT); // initialize the digital pin as an output
  pinMode(buttonPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  buttonState = digitalRead(buttonPin); // read the state input at execution
  if (buttonState == HIGH){ // if the button is pressed then...
    digitalWrite(ledPin, HIGH); // turn on the LED
  }
  else { //otherwise...
    digitalWrite(ledPin, LOW); // turn off the LED
  }
}
```

Exploration

LEDs Combined – Blink on Button Press

Let's combine all that we know with LEDs to make the LED perform a blinking pattern when the button is pressed.

In our if statement where an action is performed if the button is pressed, make the light blink 3 times with each blink lasting ½ of a second.

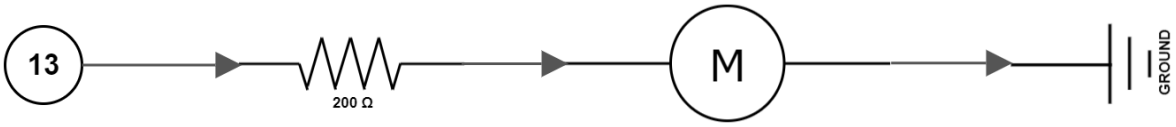
Movement – Motors

Hardware Required

- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- 220-ohmhm resistor x 1
- Car Kit that includes DC Motor (Standard 12VDC) x 4

Schematic

Circuit One:
Connect the digital pin (pin 13) to one end of the 220-ohm resistor, followed by connecting one side of the motor to the resistor and the other side to the ground.



Step-By-Step Programming

Plug in your board to the computer and start the Arduino IDE.

Initialize digital pin 13 as the pin that your motor will be connected to. Digital Pin 13 is also connected to the motor onboard the Arduino Uno which allows for easier circuit debugging, if necessary.

```
int motorPin = 13;
```

In your setup, initialize digital pin 13 to be an output pin by calling pinMode().

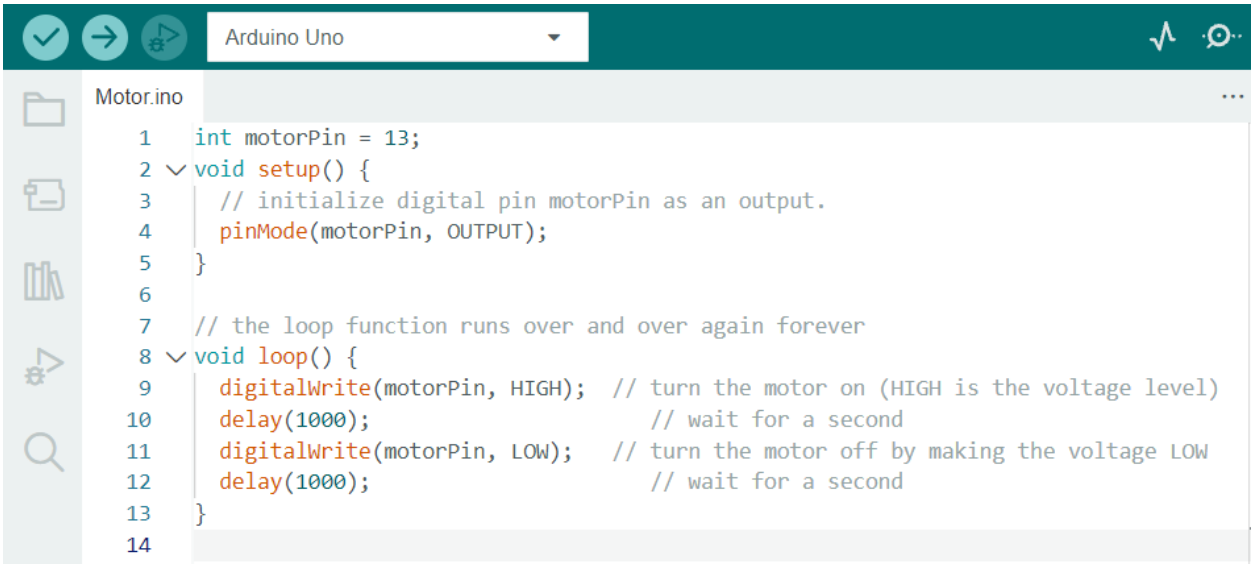
```
void setup() {
  pinMode(motorPin, OUTPUT);
}
```

In your loop, make the motor turn on and off; this means that a voltage will be provided so that the motor turns on for a specified period followed by removing the voltage provided so that the motor turns off. To do so, use digitalWrite() to declare whether the voltage is at HIGH or LOW followed by a delay() call.

```
void loop() {
  digitalWrite(motorPin, HIGH);
  delay(1000);
  digitalWrite(motorPin, LOW);
  delay(1000);
}
```

Compile your code to double-check that there are no errors and that the program itself will upload and run onto your chosen device. If the compilation is successful, upload the program and watch how the motor reacts.

Completed Code with Comments



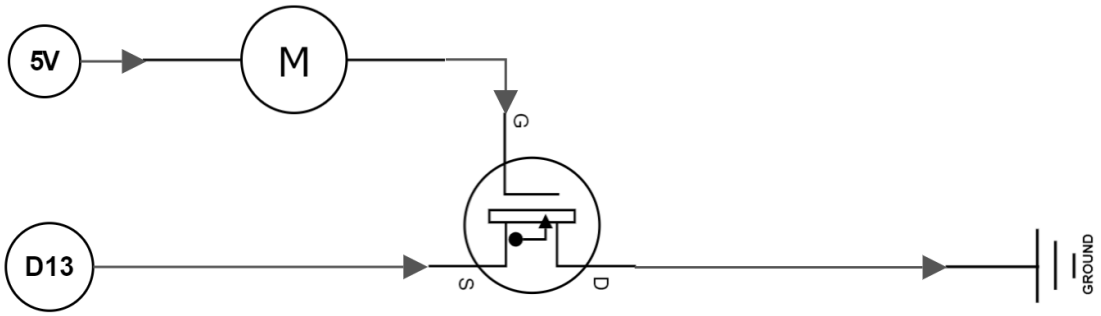
Movement – Motors and MOSFETS

Hardware Required

- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- 220-ohmhm resistor x 2
- Car Kit that includes DC Motor (Standard 12VDC) x 4
- MOSFETS x 2

Schematic

Circuit One:
Connect the leftmost pin of the MOSFET to a digital output pin (D13). The motor is connected to voltage (positive) and the middle pin of the MOSFET. The rightmost pin of the MOSFET to ground.



Step-By-Step Programming

The code remains identical to the previous lesson, *Movement – Motors*.

Exploration

How do MOSFETS work?

Connect a second motor to the first circuit's MOSFET.

Now create a second circuit like the first and connect another two motors to that second MOSFET.

Create a code in which you turn on one MOSFET and two motors for a period whilst the other is not on. Then switch the MOSFETS to allow the other two motors to move for a period.

HINT: Assign the other circuit to another pin and follow the same tactics we've used before.

What are the purposes of a MOSFET? Why would they come in handy?

_____

_____

_____

_____

One use of MOSFETs is the capability of making only connected equipment react in a certain way. You can make the cart take left and right turns by only turning on one set of motors with the MOSFET will keep the other MOSFET off.

PROJECT

Project Kit

For the following project, be sure to have the following listed below:

Required:

- Arduino Uno
- Arduino IDE
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- 220-ohm Resistors x 2
- MOSFET x 1
- Car Kit that includes DC Motor (Standard 12VDC) x 4
- Meter Stick
- Tape
- Stopwatch

Purpose

A DC motor's operating principles are reasonably straightforward. A coil is placed in a magnetic field, that's the armature placed within the permanent magnets, and a current is passed through the coil. The current passing through the coil produces torque (a rotational force) that turns the motor's shaft and, in turn, the attached load. Since the whole operation depends on applying a current to the coil of the motor, the voltage of the source power is directly related to the motor's output speed.

A DC motor's speed is directly proportional to the input voltage. The higher the input voltage, the faster the output speed. The lower the input voltage, the slower the output speed.

We can control the speed independently of torque by manipulating the supply voltage using a DC motor control unit. This allows the operator to maintain a steady torque over varying speeds or maintain a constant speed over a variable load.

Let’s explore this correlation and find out the relation between voltage and velocity.

Cart Racing – Voltage and Velocity

Hardware Required

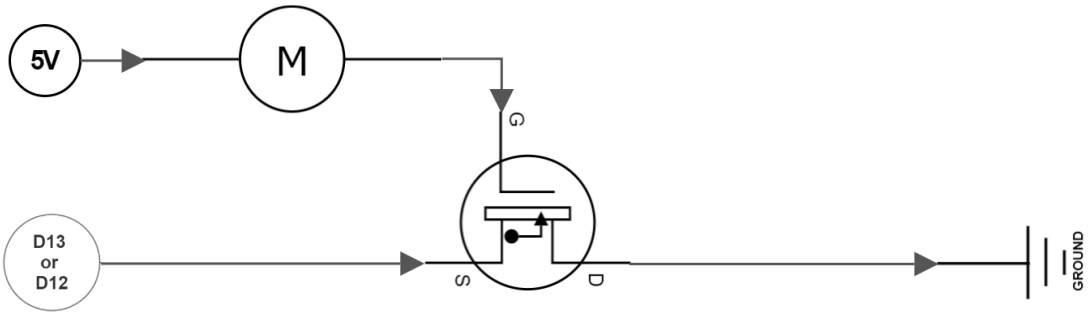
- Arduino Uno Board
- Breadboard
- Wires (Suggested 22AWG-Silicone Stranded Wire)
- 220-ohmhm resistor x 2
- Car Kit that includes DC Motor (Standard 12VDC) x 4
- Button x 1
- MOSFETS x 2
- Meter stick
- Tape
- Stopwatch
- Computer with Excel

Schematic

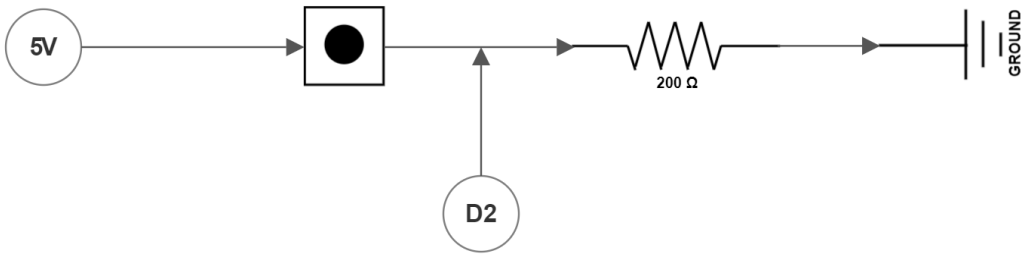
Combining various pieces of previous lessons, let's create a car that will run a program when a button is pushed the car will travel for a certain amount of time.

Take the completed circuits from the *Blink 3 – LEDs and Buttons* and *Movement – Motors and MOSFETs*.

Circuit One and Two:
Create two identical circuits with two MOSFETS that are connected to the right and left motors of your cart, respectively. Connect the leftmost pin of a MOSFET to a digital output pin (D13 or D12). The motor is connected to the voltage (positive) and the middle pin of the MOSFET. The rightmost pin of the MOSFET to ground.



Circuit Three:
Connect the rightmost pin of the button to the voltage (positive), and the leftmost pin to the digital pin (D2) and a resistor. The other end of the resistor should be connected to ground.



Step-by-Step Programming

Assign different digital pins for the two sides of the car's motor, as well as assign the button an analog pin. Once the pins are assigned, declare the digital pins as OUTPUT pins and the button as an INPUT pin.

Create an action in our loop function where we will assign the action of turning on both MOSFETs for 10 seconds once the button is pressed.
HINT: Like the code with lesson 3 but replace our LED digital pins with MOSFETs.

Test the button to see if the cart will run for 10 seconds before stopping.

Experimentation

Once the circuit and code are complete, take a meter stick and measure out 5 meters on the floor and mark it with tape. Place your cart on the starting line, and with your stopwatch, measure the time it takes for the car to travel from the starting line to the end line when your button is pressed. Perform this measurement 3 times.

Using the data that you have collected, calculate the velocity of your specific cart.

On the floor, using colored tape and a meter stick, create a track in which your cart will be able to travel automatically through your written code and measured velocity.