Sample Arduino Curriculum (All ages)
The material is not split up into days, rather sections so that progress can be made at the student's pace.

1. **Diagnostic Test**

The student will be required to provide a solution to either a programming problem or a circuitry problem, or both, based on their expertise and interests. This test is intended to be difficult, and multiple factors will be considered **regardless of whether they were able to solve the problem successfully.**

Programming-based prompts may include:

1. Write a function to reverse the order of words in a given sentence.
2. Implement a simple database using an array and create a method to add, retrieve, and delete elements.
3. Write a program to find the factorial of a given number using recursion.
4. Design a class hierarchy for a simple object-oriented system that models a university with students, courses, and teachers.
5. Implement a simple search engine that can search for keywords in a given text.
6. Write a program to convert decimal numbers to binary and vice versa.
7. Implement a sorting algorithm, such as bubble sort, selection sort, or insertion sort, and compare its performance to a built-in sorting method in your programming language.
8. Create a system of classes to model a simple economy, including banks, accounts, and transactions.
9. Implement a basic error handling system that catches exceptions and provides appropriate feedback to the user.
10. Write a program that simulates a simple game, such as Tic-Tac-Toe or Hangman, and allows a user to play against the computer.

Circuitry-based prompts may include:

1. Design a simple circuit using a breadboard that can control an LED with a push button.
2. Implement a basic analog-to-digital conversion circuit using a microcontroller and a potentiometer.
3. Create a circuit that can detect and respond to different types of inputs, such as temperature or light sensors.
4. Design a circuit that can generate different audio frequencies using a microcontroller and a speaker.
5. Implement a simple circuit that can communicate with other circuits or devices using wireless technology, such as Bluetooth or Wi-Fi.

The following will be considered:

- Proficiency: Assess the student's ability to work with the chosen language or tools, as well as their understanding of the language's features, libraries, or the principles of circuit design.

- Problem Solving: Evaluate the student's ability to break down the problem, identify the key components, and develop an effective solution, whether it's a program or a circuit design.
- Creativity: Consider the student's ability to think outside the box and come up with innovative approaches to solving the problem, as well as their willingness to explore new ideas and techniques.
- Design Quality: Assess the quality of the student's design, including factors such as readability, maintainability, and documentation for programming projects, or clarity, efficiency, and reliability for circuit designs.
- Code Management: Consider the amount of time taken to complete the project, and evaluate the student's ability to manage their time effectively.
- Integration of External Resources: Assess the student's ability to find, understand, and integrate external resources, such as libraries, online tutorials, or component datasheets, into their project.
- Error Handling and Debugging: Evaluate the student's ability to handle errors and bugs in their code or design, as well as their problem-solving skills when it comes to debugging.
- Testing and Documentation: Assess the student's ability to test their code or design thoroughly and document their project, including the purpose of the program or circuit, the main functions or components, and any potential limitations.
- Communication and Collaboration: Evaluate the student's ability to communicate their thought process, the solutions they have implemented, and any challenges they faced during the project.

2. **Introduction to Programming and Circuitry:**
   * Overview of programming languages and tools

Depending on what the student knows already, some portions may be skipped. The student will begin with using the following software:

- MIT’s Scratch

Only will be used if truly necessary. Utilizing this software for an extended period of time will hinder the programmer’s growth.

- Visual Studio Code

This IDE contains various features that will be useful throughout the programmer’s life, such as autocomplete, advanced syntax highlighting, inline compiler warnings, easy git usage, and more. The student is not required to install this, but poses a good means of education in contrast with Arduino IDE, which is tailored for the Arduino and which **is** required.

- C++

This is the language that Arduino uses to burn code to its ROM. The student will be asked to setup C++ and will be encouraged to experiment with the language. Educational materials, such as pamphlets or books, will be provided.

* Introduction to Arduino and its capabilities

The student will learn about Arduino and will learn about its applications. The student will learn how to use Arduino IDE and how to burn example programs to the board. The student can experiment with the board in their free time.

* Basic electronics and circuitry concepts

The student will learn about binary and how it is implemented using high and low voltage, how to connect the Arduino to a breadboard, and other concepts such as the ground wire. These materials will be provided if the student doesn't have them, but it is recommended to purchase an Arduino starter kit so that the student can explore out of their own curiosity.

3. **Programming Fundamentals:**

This section is intended to be one of the faster sections. If the student needs more time, the instructor will lower the pace, but in order for the student to accomplish their Arduino project goal, this section will have to be completed at a faster pace.

* Variables, data types, and operators

Simple programming concepts, such as variables, data types, strongly vs weakly typed, interpreters vs compilers, will all be taught to the student. If the student is not able to understand these fundamental data types, additional time and material will be utilized as per the instructor's discretion, because these concepts are **crucial**.

* Control structures (if-else, loops, functions, etc.)

Similar to the necessity of the previous section, time will be allocated as needed. The student will generally not face any major difficulties.

* Pointers and structs

This section is arguably the most difficult of all. The student will first conceptually understand reference vs value data and pointers and how to utilize them safely. The student will also learn about important library calls like malloc(3) to further their conceptual understanding. The student will learn about Structured Programming and how to work with typedef and structs at the lowest level. These constructs, which classify as Functional Programming, will be the most useful programming knowledge granted to the student. The student will be able to apply the concepts taught here to understand **any language** conceptually and will gain a stronger understanding of programming in general.

The application of this information is not limited to programming. Pointers and structs being low-level constructs are extremely memory and CPU efficient. The Arduino board, with limited resources, would be among the most heavily impacted by the programmer utilizing these constructs.

* Object-oriented programming (OOP) concepts

Object-oriented concepts like inheritance, polymorphism, methods, public/private/protected access, will be much easier for the student to visualize conceptually once they grasp the low level implementation of it by understanding the VMT (Virtual Memory Table) and how it is constructed off of pointers.

In this section, the student may draw parallels between Python and C++. For this reason, it is recommended to install an IDE like Visual Studio Code due to its established support for multiple languages.

4. **Circuitry and Electronics:**

This section focuses on the fundamental concepts of circuitry and electronics that are essential for working with Arduino projects. The topics covered in this section include digital and analog circuits, working with sensors and actuators, and interfacing with microcontrollers.

* Digital and Analog Circuits:
In this part, students will learn about the differences between digital and analog circuits, their components, and how they function. They will also study various digital and analog circuits, such as logic gates, flip-flops, amplifiers, and filters. Understanding these circuits is crucial for interfacing sensors, actuators, and microcontrollers.

* Working with Sensors and Actuators:
This section covers the interaction with various types of sensors and actuators commonly used in Arduino projects, such as temperature sensors, light sensors, servo motors, and stepper motors. Students will learn about the interface protocols, connection methods, and data communication between sensors and actuators with the Arduino board. They will also learn how to calibrate and configure sensors and actuators for specific applications.

* Interfacing with Microcontrollers:
In this section, students will explore the fundamentals of microcontrollers, including their memory, input/output capabilities, and programming interfaces. They will learn how to connect and communicate with the Arduino board using programming languages like C++ and Arduino's built-in libraries. Students will also gain hands-on experience in writing code to control sensors, actuators, and other peripherals connected to the Arduino board.

5. **Arduino Programming:**

This section of the curriculum focuses on the programming aspect of Arduino development. It covers the installation and setup of the Arduino Integrated Development Environment (IDE) and introduces fundamental programming concepts specific to the Arduino platform. Students will learn at a steady pace, ensuring they understand each concept before moving on to the next.

- Installing and Setting Up the Arduino IDE:
  In this part, students will learn how to download and install the Arduino IDE on their computers, as well as how to set up the environment, provided they have not already. This process will be guided step-by-step, with hands-on experience in setting up the IDE and configuring the necessary tools.

  * Basic Arduino Programming Concepts
  This section introduces students to the fundamental programming concepts used in Arduino development. Topics include the syntax of variables, functions, loops, conditional statements, and basic input/output (I/O) operations in the Arduino C++ language. Students will learn to write simple Arduino sketches (programs) and understand the purpose of each concept.

  * Working with Input/Output (I/O) Pins and Libraries
  Here, students will delve into the core functionality of the Arduino board by learning about input and output pins, digital and analog I/O, and the use of libraries. They will learn how to control and manipulate digital and analog pins, work with sensors and actuators, and utilize the built-in libraries provided by the Arduino IDE.

  * Intermediate Arduino Programming Techniques
  In this part, students will expand their knowledge of Arduino programming by exploring more advanced concepts. Topics include advanced I/O operations, interrupts, timer functions, and advanced library usage. Students will learn to enhance their sketches and develop more complex projects.

6. **Advanced Arduino Projects:**

This section builds upon the foundational knowledge gained in the previous sections and delves into more complex and sophisticated Arduino projects. Students will explore advanced topics such as designing and implementing more intricate circuits, working with wireless communication protocols, and integrating advanced sensors and actuators into their projects.

  * Designing and Implementing More Complex Circuits
  In this part, students will learn to design and build more complex circuits that may include multiple microcontrollers, sensors, and actuators. They will understand the importance of proper circuit layout, power management, and troubleshooting techniques. Students will further their understanding of breadboards, and if time and resources permit, they will work with PCBs (Printed Circuit Boards), and learn how to solder components onto the boards.

  * Working with Wireless Communication (Bluetooth, Wi-Fi)
  This section covers the integration of wireless communication technologies, such as Bluetooth and Wi-Fi, into Arduino projects. Students will learn how to set up and configure wireless modules, as well as how to transmit and receive data using these protocols. They will also explore applications and examples of wireless communication in various Arduino projects. Depending on the scope of the student’s personal project and the time remaining, this section may be briefly touched upon.

- Advanced Sensor and Actuator Integration:
  In this section, students will expand their knowledge of sensors and actuators by exploring more advanced devices and integrating them into their projects. Topics may include the use of high-precision sensors, motor control algorithms, and advanced robotics. Students will learn to choose the appropriate sensors and actuators for their projects, as well as how to configure and optimize their performance.

8. Integrating Programming and Circuitry:

This section focuses on the integration of programming and circuitry concepts to create more robust and sophisticated Arduino projects. Students will learn to seamlessly combine their knowledge of programming and circuitry to design and implement projects that utilize advanced features and technologies. They will gain a deeper understanding of how programming concepts, such as pointers, objects, and libraries, can enhance the functionality and efficiency of circuits. In turn, they will also learn how to apply circuitry skills, like working with sensors and actuators, to create more complex and versatile programming applications.

Some topics covered in this section include:

* Real-time programming and system performance optimization:
Students will explore how to optimize their code for real-time performance, ensuring their projects run smoothly and efficiently. They will learn about interrupt handling, timing mechanisms, and the use of non-blocking functions in asynchronous programming.

* Advanced sensor and actuator integration:
In this section, students will expand their knowledge of sensors and actuators by exploring more advanced devices and integrating them into their projects. They will learn how to choose the appropriate sensors and actuators for their projects, as well as how to configure and optimize their performance depending on their program.

* Wireless communication and networking:
Students will learn about wireless communication protocols, such as Bluetooth, Wi-Fi, and LoRa, and how to integrate them into their Arduino projects. They will explore the possibilities of creating connected devices and networks, and how to secure their projects against potential threats.

* Implementing security and safety features:
Students will gain knowledge on how to integrate security and safety features into their Arduino projects, including encryption, authentication, and error handling. They will learn how to protect their projects from unauthorized access and ensure the safe and reliable operation of their devices.

<u>Now that the student has the experience to create their dream project, they will begin working on it!</u>

1. Project Ideation:

If the student doesn't have any idea of what they want their project to be, or their project cannot be accomplished due to resources or limited experience, they will progress through this section.

* Students brainstorm ideas for their dream Arduino-based project, considering the skills and knowledge they have acquired throughout the course.

The instructor **cannot decide whether their project is realistic or not.** The student will undergo adequate research that is necessary for their project. The instructor is allowed to make suggestions but should not hinder the creativity of the student. This is the part where the student's experience shines.

* Students refine their ideas, breaking them down into manageable components and identifying the required hardware and software components.

If the task cannot be accomplished, the student will most likely find out during this portion of their project. If it cannot be accomplished, they will brainstorm more ideas or perhaps limit the extent of their project during this course. The student **can incorporate those features outside the course**, but that is beyond the scope of this course.

2. Project Planning:
   * Students create a detailed project plan, including a timeline, resource requirements, and milestones.

Organization is key in a complex project. The instructor will guide the student in this portion of the project. This portion will be kept to a minimum so that the student can begin with real world work. In some scenarios, the student will be recommended to write pseudocode.

* Students identify any potential challenges and develop strategies to address them.

The student may be asked to utilize visual thinking skills on art and apply them to this project in order to aid their thinking and improve their observation.

3. Design and Prototyping:
   * Students design the circuitry and software components of their project, incorporating the skills and techniques they have learned.

As a complement to the pointer section, this will be the longest --and most difficult-- section for the student. The instructor will guide the student, but the instructor will be asked to refrain from giving excessive advice, as that could lower the quality of the student's work. The student will be encouraged to use resources like StackOverflow, ChatGPT, and GeeksForGeeks as these are all real-world programming skills. The student will be recommended to create a git repo on a platform like GitHub or Codeberg.

* Students create a prototype and/or physical version of their project, testing and refining the design as needed.

This section will require purchase of hardware, unless the student and/or their parents limit the scope of the project to a simulation, which is not recommended as it will severely impact their satisfaction. The student has completed their entire project from the ground up with **minimal guidance**, the student will be heartily congratulated and will begin working on some of the more difficult features if time permits.

4. Documentation and Presentation:
    * Students document their project, including the design, code, and any challenges they faced and solutions they developed.

Code without documentation cannot be utilized properly by anyone other than the creator- the student will write formal documentation for any users or collaborators of the project. The student will document portions of the project which were difficult for the student on a public platform as a part of their learning experience and to aid future students.

    * Students present their project to their peers and instructors, highlighting the key features and demonstrating the functionality of their creation.

The student has prepared documentation, but live presentation with proper presenting skills is essential to a programmer, especially if the student is interested in pursuing programming/circuitry in college. The other students will be asked to listen and will not be required to take notes, as the presenting student should be able to captivate their audience.

5. Reflection and Improvement:
    * Students reflect on their project, identifying areas for improvement and potential future enhancements.

The student may want create a to-do list if they intend on pursuing this project out of the scope of this course, as that would aid their organization. The student will document their reflections so that they can improve their future projects whether in future courses or not.

    * Students consider how their experience with this project can be applied to future projects and their overall understanding of programming and circuitry.

The student has learned a lot along the way, and the time the student spent will grant invaluable experience. The student will apply concepts such as metacognition to improve their thinking and problem solving on a broader scale, and will be encouraged to make this a habit as it contributes to their learning experience.

The student can now take part in other courses or complete projects on their own, depending on their curiosity and their educational requirements. The instructors will be asked to complete a self reflection and the students will reflect on their instructors.

Note: This curriculum is only a sample curriculum. This may be modified or adjusted to suit the students needs, as our organization specializes in working in a student per student basis.