ROBOT DRONE LEAGUE
2024 Challenge: MINESHAFT

Standards Alignment with the Kentucky Academic Standards for Computer Science and Technology

RDL Introduction
Creativity and innovation are key elements to advancing the fields of science, technology, engineering, and mathematics (STEM) into the future. Robot Drone League (RDL) has been designed to provide students with open-ended challenges that allow for creation and innovation by engaging in hands-on design, engineering, and programming of interactive robots and drones. Students are presented with the opportunity to develop real-world connections to classroom learning. Working with robots in a collaborative game format can be a very powerful tool to engage students and enhance math and science skills through hands-on, student-centered learning. Through participation in RDL, students can develop the essential life skills of teamwork and collaboration, as well as critical thinking, project management, and communication required to become the next generation of innovators and problem-solvers in our global society.

Middle School (6-8)
Kentucky Academic Standards for Computer Science

Concept: Algorithms and Programming
Identifier Standard & Description
M-AP-01
Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
Collaboration is a common and crucial practice in programming development. Program developers often take on varying roles during the design, implementation and review stages of program development, including but not limited to graphic design and code writing.
Subconcept: Program Development
M-AP-02
Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
Decompose (break down) problems into smaller, more manageable, individual steps. Stepping through the execution of a program is a common practice when debugging and ensuring the accuracy of the program.
Subconcept: Modularity
M-AP-03
Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
Solicit diverse perspectives throughout the design process to improve artifacts. Considerations of the end-user may include usability, accessibility, age-appropriate

content, respectful language, user perspective, pronoun use, color contrast, and ease of use.

Subconcept: Program Development

M-AP-04

Create flowcharts and/or pseudocode to address complex problems as algorithms.

A flowchart visually represents an algorithm used to solve a problem. Pseudocode uses a written language. Both are means of logically thinking through a problem before actual programming begins and identify the steps needed to process input(s) to produce the desired output(s).

Subconcept: Algorithms

M-AP-05

Create clearly named variables that represent different data types and perform operations on their values.

A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs decide when and how to declare and name new variables. Determine the appropriate type and size of variable to use. Use naming conventions to improve program readability.

Subconcept: Variables

M-AP-06

Create procedures with parameters to organize code and make it easier to reuse.

Create procedures and/or functions that are used multiple times within a program to repeat groups of instructions. Define parameters within the procedure that allow for varying input.

Subconcept: Modularity

M-AP-07

Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

Control structures can be combined in many ways. Nested loops are loops placed within other loops. Compound conditional statements use two or more conditions (e.g., AND, OR, and NOT) in a logical relationship. Nesting conditionals within one another allows the result of one conditional to lead to another.

Subconcept: Control

M-AP-08

Incorporate existing code, media, and libraries into original programs, and give attribution.

Insert portions of digital media in their own programs and websites. May also import libraries and connect to web application program interfaces (APIs).

Subconcept: Program Development

M-AP-09

Systematically test and refine programs using a range of test cases.

Evaluate whether programs function as intended. Testing should be a deliberate process that is more iterative, systematic, and proactive than at lower levels. Test programs for potential errors.

Subconcept: Program Development

M-AP-10
Document programs in order to make them easier to follow, test, and debug.
Provide documentation for end users that explains their artifacts and how they function. Proper documentation aids in debugging and future program modification.
Subconcept: Program Development
M-AP-11
Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.
Consider licensing implications for their own work, especially when incorporating libraries and other resources. When considering two software libraries that address a similar need, a choice could be justified based on the library that has the least restrictive license.
Subconcept: Program Development
M-AP-12
Develop a process creating a computational artifact that leads to a minimum viable product followed by reflection, analysis, and iteration.
Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program.
Subconcept: Program Development

Middle School 6-8 Kentucky Academic Standards for Technology

Concept: Innovative Designer (ID)
Competency: Students use a variety of technologies to design and create.

Standard:
ID1. Use a variety of technologies to identify and solve authentic real-world problems.
Learning Priority:
A. Find authentic real-world problems in local and global contexts.
Indicators for grades 6-8:
1. Collaborate with others in and out of the classroom using digital tools to identify real-world problems and propose a solution that affects the local and global community.
Learning Priority:
B. Exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.
Indicators for grades 6-8:
1. Demonstrate the ability to investigate and make sense of open-ended problems using digital tools and persevere in solving them.

Standard:
ID2. Use a variety of technologies within a design process to create new, useful and imaginative

solutions.

Learning Priority:

A. Know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

Indicators for grades 6-8:

1. Explore and choose appropriate processes and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

Learning Priority:

B. Select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.

Indicators for grades 6-8:

1. Investigate and use meaningful digital tools to plan and manage a design process that considers design constraints and calculated risks.

Learning Priority:

C. Develop, test and refine prototypes as part of a cyclical design process.

Indicators for grades 6-8:

1. Create, develop and test prototypes; understand and appreciate that failures are opportunities for growth and improvement.

Concept: Computational Thinker (CT)

Competency: Students understand sequences and use them to develop solutions to problems.

Standard:

CT1. Develop and employ strategies for understanding and solving problems in ways that leverage the
power of technological methods to develop and test solutions.

Learning Priority:

A. Formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.

Indicators for grades 6-8:

1. Ask questions, gather data, create/observe abstract models, and think of different processes while finding solutions to real-world problems.

Learning Priority:

B. Collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.

Indicators for grades 6-8:

1. Solve problems and make decisions by collecting data or identifying relevant data sets, using digital tools ex.: sheets, surveys to analyze the data, and represent their findings through various ways.

Learning Priority:

C. Break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

Indicators for grades 6-8:

1. Break problems into parts, extract key information, and develop descriptive models to understand complex systems or lead problem solving tasks.

Learning Priority:

D. Understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

Indicators for grades 6-8:

1. Use digital tools to collect data, conduct analysis, and discuss findings or possible solutions.

Standard:

CT2. Apply strategies for understanding and solving problems by using technological methods to develop and test solutions.

Learning Priority:

A. Use resources to collect, analyze, and represent data.

Indicators for grades 6-8:

1. Use digital tools to ask questions to an audience and digitally collect data, and analyze the findings.

Learning Priority:

B. Deconstruct components to understand systems and facilitate problem-solving.

Indicators for grades 6-8:

1. Use technology-assisted methods to break problems down into smaller, more manageable parts by finding patterns or other methods of decomposition.

Learning Priority:

C. Create and test automated solutions.

Indicators for grades 6-8:

1. Use algorithm design to develop step-by-step instructions for solving a problem.

Concept: Global Collaborator (GC)

Competency: Students use digital tools to connect with learners inside and outside of their classroom.

Standard:

GC1. Use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.

Learning Priority:

A. Use digital tools to connect with learners from a variety of backgrounds and cultures, engaging with them in ways that broaden mutual understanding and learning.

Indicators for grades 6-8:

1. Use digital tools and resources to connect and collaborate with authentic audiences from various backgrounds and cultures to broaden mutual understanding and learning, while using appropriate digital citizenship skills.

Learning Priority:

B. Contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.

Indicators for grades 6-8:

1. Select and use digital tools in diverse collaborative teams within the classroom, assuming specific roles, responsibilities, and perspectives other than your own, to contribute effectively toward a common goal.

Learning Priority:

C. Contribute to the exchange of ideas within and beyond the learning community.

Indicators for grades 6-8:

1. Select and use digital tools in diverse collaborative teams outside the classroom, assuming specific roles, responsibilities, and perspectives other than their own, to contribute effectively toward a common goal.

Standard:

GC2. Use digital tools to connect with a global network of learners and engage with issues that impact local and global communities.

Learning Priority:

A. Use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.

Indicators for grades 6-8:

1. Use collaborative technologies to connect with others - including peers, experts, and community members - to learn about issues and problems or to gain diverse local and global perspectives.

Learning Priority:

B. Explore local and global issues and use collaborative technologies to work with others to investigate solutions.

Indicators for grades 6-8:

1. Use collaborative technologies and assume roles within digital creations while maintaining digital citizenship within the team digital workspace to investigate and develop solutions to local and global issues.

High School (9-12)
Kentucky Academic Standards for Computer Science

Algorithms & Programming
Identifier Standard & Descriptor
H-AP-01
Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.

At previous levels, students adhered to licensing schemes. At this level, students should consider and be able to explain licensing implications for their own work, especially when incorporating libraries and other resources.

Subconcept: Program Development
H-AP-02
Use a development process in creating a computational artifact that leads to a minimum viable product followed by reflection, analysis, and iteration.

At previous levels students have developed artifacts in ad-hoc way. By high school,

students should be introduced to the discipline of software development. In particular, we focus on analysis, reflection and iteration. When given a problem students should be able to fully explain the problem, consider possible ways to solve it and then apply one of the possible ways. Consideration of possible ways to solve a software problem is termed 'Software Design' and/or 'Analysis'. Once the solution has been implemented and tested, students should reflect on what worked and didn't work during that process. Students will then apply this process again to update the artifact (iteration), continuing to refine the artifact itself while also continuing to improve the process.

Subconcept: Program Development

H-AP-03

Use functions, data structures or objects to simplify solutions, generalizing computational problems instead of repeated use of simple variables.

Students should be able to identify common features in multiple segments of code and substitute a single segment/abstraction (function, data structure or object) to account for the differences.

Subconcept: Variables

H-AP-04

Design and iteratively develop event-driven computational artifacts for practical intent, personal expression, or to address a societal issue.

Relevant computational artifacts include programs, mobile apps, or web apps. Events can be user-initiated, such as a button press, or system-initiated, such as a timer firing. At previous levels, students have learned to create and call procedures. Students should be able to design and implement procedures that are called by events.

Subconcept: Program Development

H-AP-05

Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

Students should be able to decompose complex problems into manageable sub problems that could potentially be solved with programs or procedures that already exist.

Subconcept: Modularity

H-AP-06

Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance and explain the benefits and drawbacks of choices made.

Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures should include conditional statements, loops, and event handlers.

Subconcept: Control

H-AP-07

Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

A prototype is a computational artifact that demonstrates the core functionality of a product or process. Prototypes are useful for getting early feedback in the design process, and can yield insight into the feasibility of a product. The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm.

Subconcept: Algorithms

H-AP-08

Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.

Subconcept: Program Development

H-AP-09

Evaluate and refine computational artifacts to make them more usable and accessible using systematic testing and debugging.

Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

Subconcept: Program Development

H-AP-10

Systematically design and develop programs for broad audiences by incorporating feedback from users.

Students at lower levels collect feedback and revise programs. At this level, students should do iterations through a systematic process that includes feedback from broad audiences. It is important for students to be able to gather feedback from the audience including peers, teachers and family members to make design decisions based on the feedback.

Subconcept: Program Development

H-AP-11

Design and develop computational artifacts working in team roles using collaborative tools.

As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students. Students might work as a team to develop a mobile application that addresses a problem relevant to the school or community, selecting appropriate tools to establish and manage the project timeline; design, share, and revise graphical user interface elements; and track planned, in progress, and completed components.

Subconcept: Program Development

H-AP-12

Describe how artificial intelligence drives many software and physical systems.

Artificial Intelligence, unlike algorithms, is a mechanism where programs make decisions on what to execute based on its environment similar to how humans make decisions. In the era of big data, artificial intelligence is becoming pervasive and students should be able to describe how AI is used in everyday software systems.

Subconcept: Program Development

H-AP-13

Use and adapt classic algorithms to solve computational problems.

Students should be able to identify and use well-known algorithms in sorting (e.g., bubble sort, quicksort, merge sort, insertion sort), searching (e.g., linear search, binary search), and shortest-path (e.g., Dijkstra's algorithm) problems. Students will also be able to adapt and combine such well-known algorithms to add features that address more complex computational tasks.

Subconcept: Algorithms

H-AP-14

Evaluate algorithms in terms of their efficiency, correctness, and clarity.

Students should be able to calculate the total number times a loop will be executed given a code snippet, will be able to state whether an algorithm is correct for solving a given problem, and compare/contrast algorithms for clarity and the number of executed operations.

Subconcept: Algorithms

H-AP-15

Compare and contrast fundamental data structures and their uses.

Students should be able to name the fundamental data structures (array, list, stack, queue and tree) and defend the use of a data structure's use to solve different problems in sorting and searching.

Subconcept: Control

H-AP-16

Illustrate the flow of execution of a recursive algorithm.

A recursive algorithm is a procedure or function which when implemented in a programming language calls itself. The algorithm solves a smaller problem with each call. Students should be able to identify the termination case and recursive call case in a recursive algorithm and describe the state of the problem space during each call.

Subconcept: Algorithms

H-AP-17

Construct solutions to problems using student-created components, such as

procedures, modules and/or objects.
At this level, students should be regularly implementing programming solutions using some form of structured design with multiple functions/procedures/modules in different files. Object-oriented programming is optional at this level. Problems can be assigned or student-selected.
Subconcept: Program Development
H-AP-18
Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
As students encounter complex, real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable sub problems that could potentially be solved with programs or procedures that already exist.
Subconcept: Modularity
H-AP-19
Select and employ an appropriate component or library to facilitate programming solutions.
Students should be able to use (or, actually reuse) existing code when quality code already exists to accomplish the needed task. Libraries and APIs can be student-created, part of the software development platform or external libraries or APIs selected for their features.
Subconcept: Program Development
H-AP-20
Develop programs for multiple computing platforms.
Students need to understand the pervasiveness of computing and that computers exist in many different forms. Students should be able to develop software programs that run on a desktop/laptop as well as other IOT and/or mobile devices.
Subconcept: Program Development
H-AP-21
Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.
Students need to learn the major tools and skills of software development including version control systems to keep track of all releases (and to back out changes), IDEs to simplify writing and testing of code, documentation of code for code maintenance. Students should be able to explain how these tools are critical to team-developed projects. Group software projects can be assigned or student-selected.
Subconcept: Control
H-AP-22
Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., introducing errors).
Students should be able to understand and provide examples highlighting that adding functionality to software can introduce new errors, reduce functionality or add an unintended feature.
Subconcept: Program Development
H-AP-23
Evaluate key qualities (including correctness, usability, readability, and efficiency)

of a program.
Given a software program, students should be able to evaluate it in terms of software quality using standard software metrics including readability, usability and efficiency.
Subconcept: Program Development
H-AP-24
Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.
Students should be able to explain the difference between a compiled and scripted programming language, defend a choice of a programming language for a certain computing device and defend a choice of a language (3rd generation versus 4th generation) for solving different types of problems.
Subconcept: Program Development

High School 9-12 Kentucky Academic Standards for Technology

Concept: Innovative Designer (ID)
Competency: Students use a variety of technologies to design and create.
Standard:
ID1. Use a variety of technologies to identify and solve authentic real-world problems.
Learning Priority:
A. Find authentic real-world problems in local and global contexts.
Indicators for grades 9-12:
1. Use a variety of technologies to independently identify real-world problems in the local and global community.
Learning Priority:
B. Exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.
Indicators for grades 9-12:
1. Use a variety of technologies to independently demonstrate perseverance when dealing with ambiguous and open-ended problems.
Standard:
ID2. Use a variety of technologies within a design process to create new, useful and imaginative solutions.
Learning Priority:
A. Know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
Indicators for grades 9-12:
1. Self-select and use a variety of digital tools within a deliberate process for generating ideas, researching, and testing ideas for solving problems or creating original products that demonstrate understanding.
Learning Priority:
B. Select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
Indicators for grades 9-12:

1. Self-select and use appropriate digital tools to manage work and create original products that take into consideration project constraints, obstacles and outcomes.

Learning Priority:

C. Develop, test and refine prototypes as part of a cyclical design process.

Indicators for grades 9-12:

1. Select and use a variety of digital tools to aid in working collaboratively or independently to create, test and refine prototypes, drafts and concepts based on self-initiated feedback and reflection in design cycles.

Concept: Computational Thinker (CT)

Competency: Students understand sequences and use them to develop solutions to problems.

Standard:

CT1. Develop and employ strategies for understanding and solving problems in ways that leverage the
power of technological methods to develop and test solutions.

Learning Priority:

A. Formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.

Indicators for grades 9-12:

1. Precisely define a problem and develop a solution using digital tools, conducting data analysis, abstract models, and algorithmic thinking.

Learning Priority:

B. Collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.

Indicators for grades 9-12:

1. Use digital tool s to effectively collect, organize, and manipulate data to test, verify, and present possible solutions to a problem.

Learning Priority:

C. Break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

Indicators for grades 9-12:

1. Evaluate the problem-solving process to deconstruct data and information to develop effective solutions to real-world problems.

Learning Priority:

D. Understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

Indicators for grades 9-12:

1. Demonstrate their understanding of automation and logic to develop a process to create and verify automated solutions.

Standard:

CT2. Apply strategies for understanding and solving problems by using technological methods to develop and test solutions.

Learning Priority:

A. Use resources to collect, analyze, and represent data.

Indicators for grades 9-12:

1. Use digital tools to collect relevant data, conduct analysis, and prepare data for presentation to facilitate problem-solving and decision-making.

Learning Priority:

B. Deconstruct components to understand systems and facilitate problem-solving.

Indicators for grades 9-12:

1. Use technology-assisted methods to more easily identify key information by breaking down data to facilitate problem-solving.

Learning Priority:

C. Create and test automated solutions.

Indicators for grades 9-12:

1. Use digital tools and algorithmic thinking to develop automated systems to test solutions.

Indicators for grades 9-12:

1. Use digital tools to collect relevant data, conduct analysis, and prepare data for presentation to facilitate problem-solving and decision-making.

Concept: Global Collaborator (GC)

Competency: Students use digital tools to connect with learners inside and outside of their classroom.

Standard:

GC1. Use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.

Learning Priority:

A. Use digital tools to connect with learners from a variety of backgrounds and cultures, engaging with them in ways that broaden mutual understanding and learning.

Indicators for grades 9-12:

1. Evaluate and use digital collaboration tools to connect with others from a variety of local and global backgrounds/cultures in order to exchange ideas, develop an understanding of diverse perspectives and encourage learning.

Learning Priority:

B. Contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.

Indicators for grades 9-12:

1. Use digital tools to contribute to a project team, determine their role and responsibility within the group and work toward a common goal or a solution to a problem.

Learning Priority:

C. Contribute to the exchange of ideas within and beyond the learning community.

Indicators for grades 9-12:

1. Select digital tools to share and exchange interests, ideas and experiences with others from within and beyond the local learning community.

Standard:

GC2. Use digital tools to connect with a global network of learners and engage with issues that impact local and global communities.

Learning Priority:

A. Use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.

Indicators for grades 9-12:

1. Use collaborative technologies to work with others peers, experts, community members to gain knowledge about issues through various perspectives and opinions and to find solutions for social change.

Learning Priority:

B. Explore local and global issues and use collaborative technologies to work with others to investigate solutions.

Indicators for grades 9-12:

1. Explore and analyze local and global issues and use collaborative digital tools to investigate, develop a plan and recommend solutions.