OzoBlockly Editor Challenge:

# ORA AI & Robotics Density Challenge

## Grades 9-12

Teacher Guide

*by Richard Born, Associate Professor Emeritus, Northern Illinois University*

***Introduction***

Welcome to the ORA AI & Robotics Density Challenge! This engaging lab is designed to immerse your students in the intersection of robotics, computer vision, and artificial intelligence. Through hands-on experience, learners will tackle a sorting problem based on the concept of density, engaging with the ORA cobot as they identify and classify five seemingly identical 3D-printed cubes. Each cube has varying internal infill settings that contribute to its unique mass and density, providing a practical exploration of fundamental physical principles.

Students will utilize a digital balance and a camera to capture and analyze the mass of each cube, leveraging a Large Language Model (LLM) to process visual data for accurate readings. The challenge not only emphasizes scientific problem-solving but also mirrors real-world applications in manufacturing, where collaborative robots enhance efficiency in various tasks.

As an educator, facilitating this lab enables you to guide students in applying theoretical knowledge to practical situations, reinforcing critical thinking, collaboration, and technology skills. Let's inspire the next generation of innovators to explore the dynamic field of robotics and AI.

If you have not already done so, you should now read through the *Student Guide* for this challenge for details on what is expected from the students. There are also two short videos that accompany this lab. It is suggested that you view them after reading through the student guide.

***Example Solution to this Challenge***

The solution to this challenge provides the opportunity for the students to develop an OzoBlockly Editor program that is modular, dividing the program logic into functions each having a specific purpose. In the solution presented here, variable and function names have been created in a way that promotes self-documentation.

Figure 1 shows the main program. It begins by calling an initialization function that takes care of preliminary house-keeping items. Then a loop is repeated five times, once for each of the five cubes whose densities are to be determined. This loop calls four function in sequence. The first function picks up the cube from its initial location. The second function moves the cube to the balance scale. The third function reads the balance, and the final function moves the cube to the proper water bowl—either the floaters bowl or the sinkers bowl.

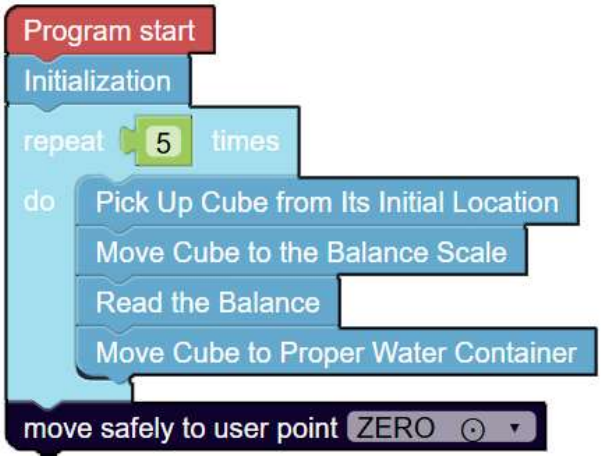


*Figure 1*

Figure 2 shows the initialization function. The cube size is set to 2.54 cm (1 inch). The volume in cubic centimeters is then calculated. The first Y pick up location is set to -110 mm. Then the increment for each additional Y pick up location is set to 55 mm, as the cubes are picked up from left to right on the grid that marks the locations for the five cubes. ORA is moved to the HOME location, and its speed is set to 20% for safety. The tool is set as the finger gripper and is initially off.

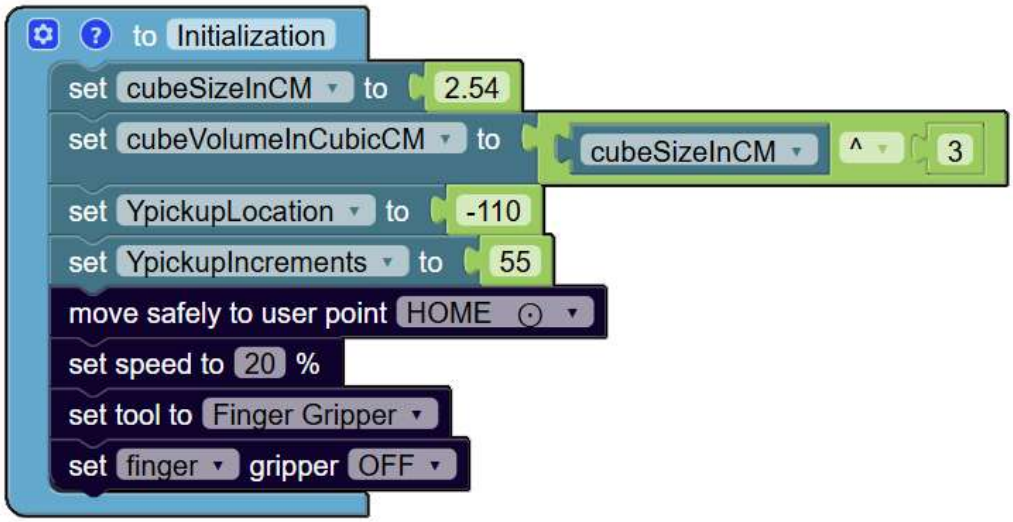


*Figure 2*

Figure 3 shows the blocks making up the "*Pick Up Cube from Its Initial Location*" function. The finger gripper is set to OPEN. The move block that follows moves the gripper to (x, y, z) = (150, YpickupLocation, 80). Note that all blocks have an initial x location of 150 mm. The next move block simply lowers the gripper so that it is at a level z = 20 mm where it can grip the block. The finger gripper is then closed, and the final move block lifts the block back up to z = 80 mm. To prepare things for the next block, the *YpickupLocation* is incremented by *YpickupIncrements* mm.

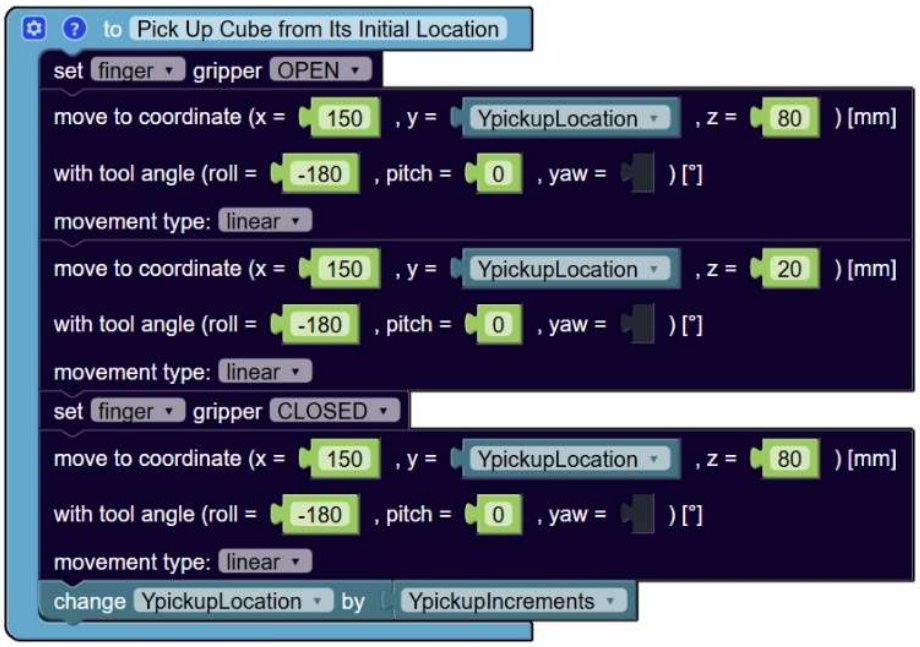


*Figure 3*

Figure 4 shows the blocks making up the "*Move Cube to the Balance Scale*" function. The center of the balance scale is located at (x,y) = (280, 0). For the first move the cube is at a height of 80 mm above the center of the scale. The second move lowers the cube to a height of 65 mm, just a couple of millimeters above the pan on the balance. The finger gripper is then opened dropping the cube to the balance. Finally, the gripper is turned OFF.

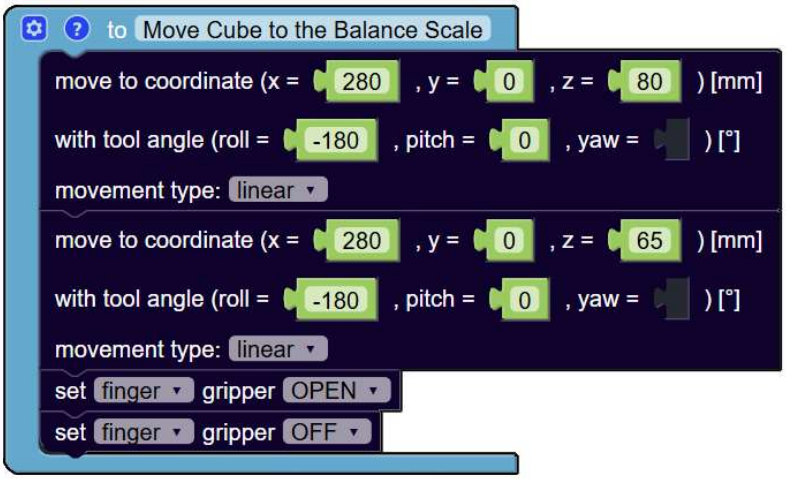


*Figure 4*

Figure 5 shows the blocks making up the "*Read the Balance*" function. The LLM is prompted for a number that is seen by the camera, *i.e.*, the *CubeMassInGrams*. The density is calculated using the physics defining formula for density, *Density = Mass / Volume*. The two *print to IO console* blocks are helpful while debugging the program but can be removed for the final program if desired.

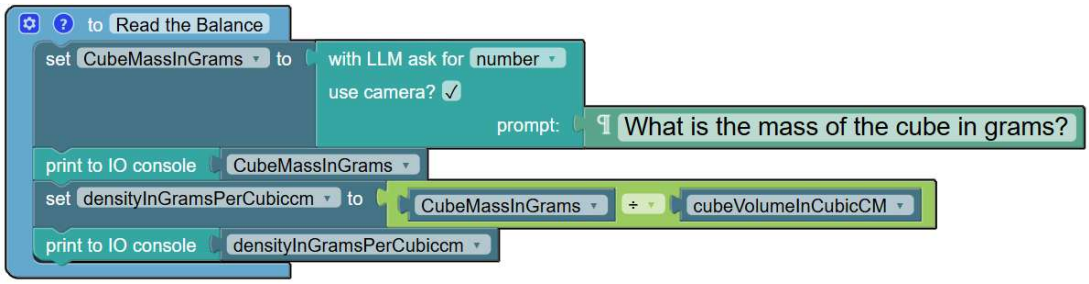


*Figure 5*

Figure 6 shows the blocks making up the "*Move Cube to Proper Water Container*" function. This function begins with an *if-do-else* block that determines which bowl to drop the cube. Floaters, with density less than 1, are dropped at (x,y) = (225, -225). Those with densities greater than 1 are dropped at (x,y) = (225, 225). The finger gripper is closed followed by a 1 second wait to ensure that the following *move* block doesn't begin before the grip is completed.

The first move block lifts the cube off the scale to a height of 80 mm above the scale. The second *move* block moves the cube to a height of 80 mm above its *yDropLocation*. The third move block moves the bottom of the cube so that it is at 65 mm, just a couple of mm above the surface of the water so that it won't splash when dropped. The finger gripper is opened, and the cube falls into the water. Finally, the gripper is turned off, and ORA is moved to the HOME position.

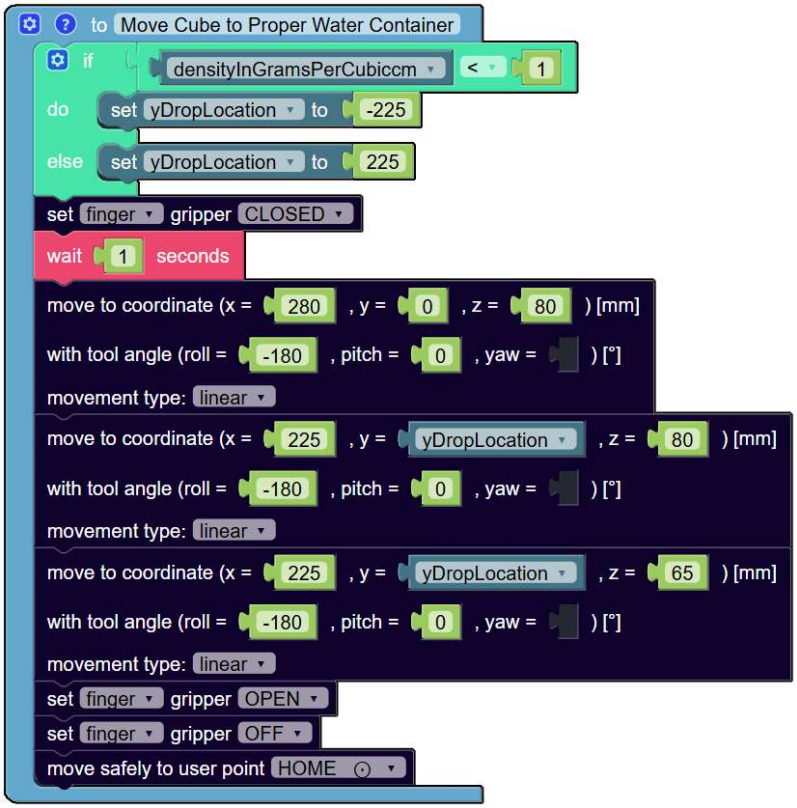


*Figure 6*

***Share Code for this Lesson – dsv7bxg***

***Zip File - Five One Inch Cubes with Varying Densities***

***ISTE Standards (International Society for Technology and Education)***

*Empowered Learner 1.1.d:* Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

*Knowledge Constructor 1.3.d:* Students build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.

*Computational Thinker 1.5.c:* Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

***CSTA (Computer Science Teachers Association) Standards***

| | |
|---|---|
| 2-CS-02 | Design projects that combine hardware and software components to collect and exchange data. |
| 2-AP-11 | Create clearly named variables that represent different data types and perform operations on their values. |
| 2-AP-13 | Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. |
| 2-AP-19 | Document programs in order to make them easier to follow, test, and debug. |
| 3A-DA-11 | Create interactive data visualizations using software tools to help others better understand real-world phenomena. |
| 3A-AP-17 | Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. |
| 3B-CS-02 | Illustrate ways computing systems implement logic, input, and output through hardware components. |
| 3B-AP-08 | Describe how artificial intelligence drives many software and physical systems. |
| 3B-AP-14 | Construct solutions to problems using student-created components, such as procedures, modules and/or objects. |