OzoBlockly Editor Challenge:

Intelligent Palletizing: Vision Guided Sorting with LLM Logic

Grades 9-12

Teacher Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

In this lesson, students will tackle this OzoBlockly Editor Challenge by programming the ORA cobot to sort playing cards into a specified 4x13 grid, mirroring essential tasks in industrial automation. Through this challenge, learners will enhance their understanding of coordinate systems, path planning, and logic-based sorting while employing a vision system to identify the cards. This exercise simulates real-world applications such as kitting and order fulfillment in warehouses, where robots organize inventory into designated bins. The project also draws parallels to palletizing and electronics assembly, where accuracy in placement is crucial. By engaging in this practical project, students will develop important automation engineering skills, culminating in a comprehensive understanding of robotics applications in various fields. Printable resources and an STL file for a 3D-printed card holder are provided to facilitate the learning experience.

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.

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. Figure 1 shows the main program, which simply calls the functions in the required order. As can be seen, it is assumed that six playing cards will be identified and moved to the 13x4 grid.

Figure 1

Figure 2 shows the initialization function. First, some ORA initialization is carried out. The vacuum gripper is set as the tool and turned OFF. ORA's speed is set to 50%, though it might be wise for students to set it lower while debugging the program. ORA is set to the HOME position.

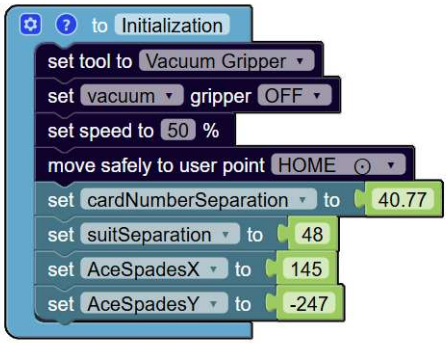

Figure 2

Next, four variables that will be used by ORA to place the card in the correct location on the 13x4 grid are initialized. Figure 3 provides a visual representation of these four variables.

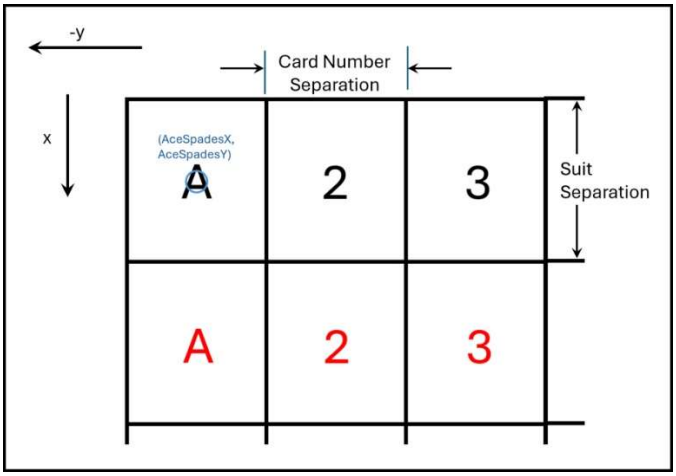

Figure 3

This portion of the grid lies in ORA's *-y* and *+x* quadrant. The center of the Ace of Spades location, represented by the center of the blue circle, is at coordinates (*AceSpadeX, AceSpadesY*). These coordinates will depend upon the exact location where the grid is placed in front of ORA. The values for the *suitSeparation* and *cardNumberSeparation* variables can be measured with a ruler and are used to determine the ultimate location to place each of the cards.

Figure 4 shows the "Identify the Playing Cards" function. A carefully worded prompt for the LLM with camera was designed to make programming as easy as possible. For example, if the camera sees the 8 of Clubs, the *LLMresponse* will be the string *3 08*. Similarly, if the camera sees the Queen of Diamonds, the *LLMresponse* will be the string *4 12*. Index 0 in the string will always be the suit number 1, 2, 3, or 4. Indexes 2 and 3 in the string will always be the card number, 1 through 13, within the suit. **Text** blocks can then be used to parse the *suitNumber* and *cardNumber,* while using **convert** blocks to convert the strings to numbers. The **print to IO console** blocks are optional, but useful, particularly when debugging the program code.

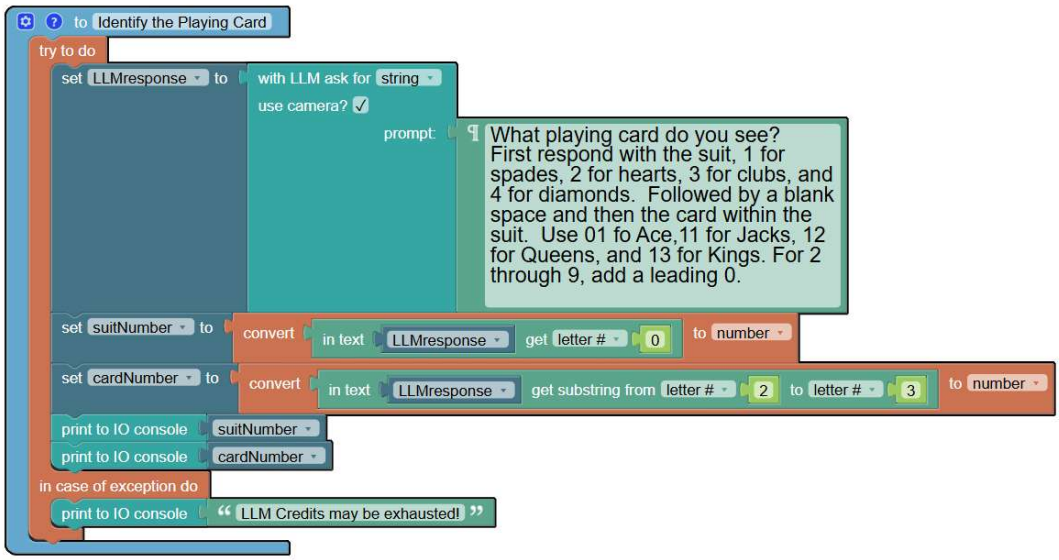

Figure 4

Figure 5 shows the "Move Card to Grid" function. The first **move to coordinate** block moves the gripper to the location of the card tray. The vacuum is turned on, the card is raised so that the gripper is above the tray. The *xPlaceLoction* and *yPlace Location* are then calculated. The card is moved to that location, lowered and then released by setting the vacuum gripper to OFF. Finally, the ORA arm is moved to its HOME position.

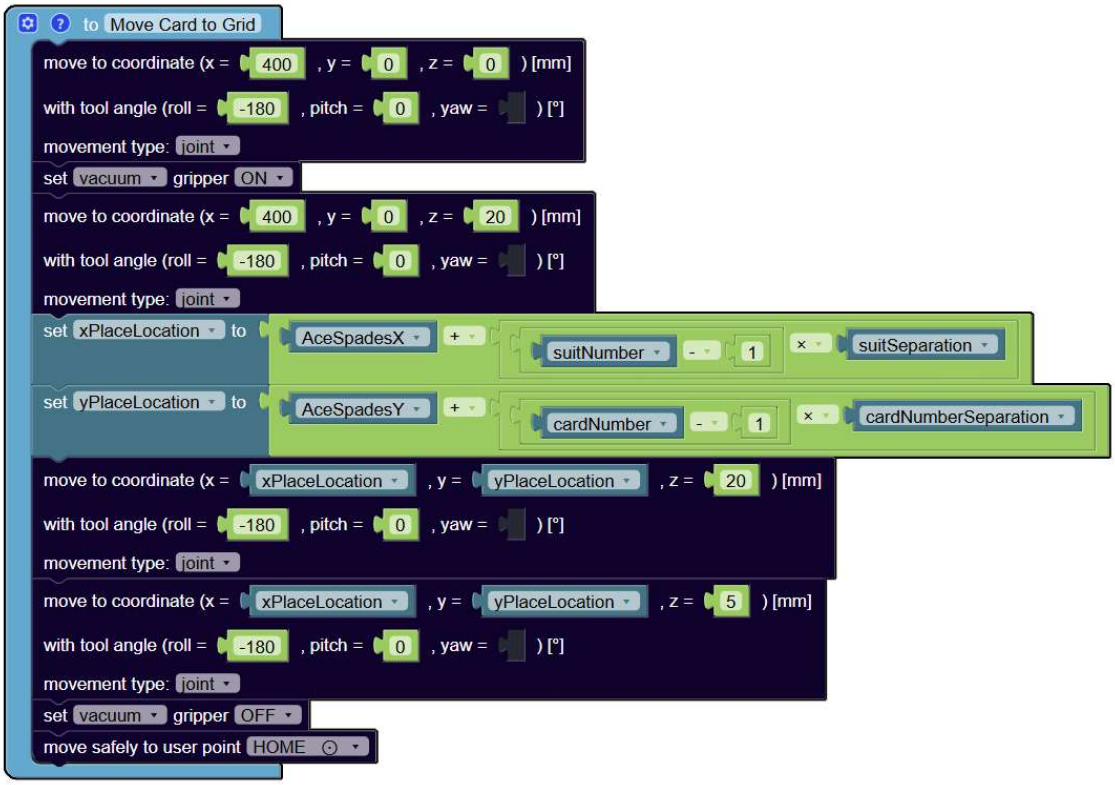

Figure 5

Share Code for this Lesson – eby5fb4

Some Helpful Suggestions

Don't expect perfection. There may be occasional mistakes where the vision system and LLM incorrectly identify a card, but probably not more than about 10% of the time. It helps if the lighting on the tray is bright and the camera is adjusted so that the card is not too small in its field of view.

The small playing cards are not very heavy and may have a tendancy to "stick" when the vacumm is turned off. It was found that using double-stick tape to tape a second card to the back of each card, making them twice as heavy, avoided this issue.

Don't try to sort an entire deck of playing cards into their respective locations in the grid. Remember that each card requries camera and LLM use as they work together. This process uses up "LLM credits". The goal is to accomplish precision placement of some cards, not trying to do this for an entire deck of playing cards.

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.