RoboTunes: A Multi-Player Learning Framework with Musical Programmable Robots

1st Khalil Al-rahman Youssefi*
Institute of Networked and Embedded Systems
University of Klagenfurt
Klagenfurt am Wörthersee, Austria
Khalil.Youssefi@aau.at
*Corresponding author

2nd Helmut Lindner
CodeLabs GmbH
Graz, Austria
hl@codelabs.at

3rd Wilfried Elmenreich
Institute of Networked and Embedded Systems
University of Klagenfurt
Klagenfurt am Wörthersee, Austria
Wilfried.Elmenreich@aau.at

***Abstract*—This paper proposes a novel learning framework that merges robotics and musical elements to create an engaging educational platform. With the increasing importance of robots in educational systems, designing a system that utilizes music and robotics as tools for teaching and learning is of high value. The concept involves a Colorized Music-based Maze (CMM) that integrates musical notes to create a challenging playground for robots. Students can instruct their robots to move on this playground utilizing a musical instrument (e.g., a piano or a flute). The proposed system offers a multi-faceted learning approach suitable for a broad spectrum of age groups, fostering collaborative learning and problem-solving skills in an interactive and enjoyable manner.**

***Index Terms*—Educational Robotics, Educational Gaming, Interactive Robotic Systems, Robotic Programming**

I. Introduction

In recent years, robots have become increasingly important to the educational systems of many countries. As technology advances, educational programmable robots are becoming more advanced and capable of performing complex tasks. Hence, they are valuable tools in teaching students various skills and concepts. The goal of this paper is to propose an educational programmable robotic-musical platform that can facilitate learning for students of different age groups. Designed robots will interact with students and help them understand musical concepts and practice skills in a fun and engaging way.

In the proposed framework, robots react (e.g., move or change status) to specific music notes. Hence, students (i.e., players) can control them by playing these notes based on their intentions. Supervisors can define tasks based on the age and abilities of students. As a simple task, for example, it can be to play a few musical notes (e.g., with a piano) in the correct order to control a robot to follow a specific path. Conversely, an advanced task can be to ask students to program a robot for the same goal. The proposed platform is designed to make the learning process more enjoyable by being used in a challenging multi-player game where individual players have distinct goals. This is an exciting new way to learn, as it has elements of fun and competition. In the proposed system, players can collaborate and compete to learn new topics while also enjoying the interactive musical aspects of the game. It's a unique and exciting way of entertaining and learning and can be a great way to make education more enjoyable.

The rest of the paper is structured as follows. Section II examines the existing hardware of available educational robots and various methods available for programming them. Section III introduces the proposed framework and a demonstrative implementation, including a hardware proposal for the intended robot. And finally, Chapter IV concludes the document.

II. Related work

Educational robots are becoming increasingly popular for educational purposes, as they provide a unique opportunity to teach kids about coding, robotics, and problem-solving in a fun and engaging way [1], [2]. This section examines the existing hardware for educational robots by providing an overview of their features and reviews different programming methods that can be utilized with educational robots.

A. Existing hardware for educational robots

Code & Go Robot Mouse is a stand-alone educational programmable robot that supports an on-robot and screen-free programming method. This robot is suitable for 4-year-old or older kids with no programming experience. Bee-bot is a similar educational robot to Code & Go Robot Mouse but in a more attractive shape. This robot was later extended to Pro-bot with more sensors and abilities.

Pro-bot is an extension to the Bee-bot, including a touch sensor on its front side, a pen that can draw lines as the robot moves, LEDs as car lights, and a speaker that can play a horn sound. The most significant improvement is an LCD screen that lets programmers see the robot's program directly on the robot itself.

Blue-bot is an educational robot that can be programmed in two different ways. The simple way to program the robot is by utilizing its onboard command keys, similar to Bee-bot. Younger kids can use this method. The second method to program this robot is to use a tactile reader. A tactile reader is a box where the programmer can place cards (program instructions) in the desired order and create a program. This method

can be used by older kids who can understand the relationship between the tactile reader and the robot. Moreover, this robot can be remotely controlled using smartphones.

Edison is another programmable robot designed to be a complete teaching resource for coding and robotics education for students ranging from 4 to 16 years old. The robot can be programmed in various ways, including reading barcodes and block-based programming.

Thymio [3]–[5] is an open-source educational robot designed by researchers from the EPFL in collaboration with ECAL and produced by Mobsya, a nonprofit association whose mission is to offer comprehensive, engaging STEAM journeys to learners of all ages. Artie Max is another robot that translates codes into colorful drawings. The design is for kids above eight years old. Several models of this robot are designed with different numbers of pens and prices.

Ozobot Evo [6] is a tiny robot (i.e., approximately 3 cm). It can be controlled remotely using a smartphone or colored lines (i.e., color codes), as the robot is equipped with a color sensor placed underside. The following functionalities and utilities can be utilized for programming or controlling: movement of the robot, audio output, proximity sensor, LED lights.

Moreover, this robot can be programmed using OzoBlockly, a customized version of open-source Blockly [7] software developed by Google. This flexibility in programming makes the bot ideal for use in long-term educational programs and for both young children and experienced students.

Sphero robot is a small, round-shaped robot. It consists of a plastic shell that encases the robot circuit, consisting of gyroscopes, motors, speakers, and LEDs. This robot is similar to the Ozobot Evo and can be controlled remotely. However, Sphero has no color sensor and can not read color-coded lines. There is also a customized version of Blockly to program the robot: the Sphero Edu application. An older version of the robot featured a type of facial recognition technology that was removed from the product due to technical problems.

Spiderino [8], [9] is another low-cost robot designed for swarm robotics and education. Its design considerations make it an ideal tool for both swarm robotics experiments and educational settings. Since it is a customizable open-source robot, various sensors can be attached, making it suitable for different applications.

B. Programming models for educational robots

Programming methods for educational robots typically include Hardware-based programming, Visual Programming, Block-based programming, and Text-based programming. This subsection briefly describes them.

For hardware-based programming, there are instruction-input buttons on robots, which programmers (e.g., students) can press sequentially to create a program. Supporting instruction-input buttons, play, pause, and stop buttons are available on the robot to control the flow of the program (e.g., Code & Go Robot Mouse and Pro-Bot). The advantage of Hardware-based programming is that there is no need for an external programmer (i.e., a laptop). However, the noticeable disadvantage is the static complexity level, which limits the reusability of the robot in educational programs.

In visual tagging, programmers (i.e., students) create a list of chosen instructions from a limited set of commands by putting them visually in order. A simple example of this programming method is the usage of a line-following robot. In this case, students draw lines that robots should follow, and those lines are programs. A more complex example in this topic is a programming method for the Ozobot Evo bot. This robot includes a color sensor that can detect line colors and follow instructions based on the color of the currently following line. Another type of this programming method is used in the Edison bot that scans barcodes to learn a program.

Programming languages are typically designed to be used by engineers and are not easy to use. Block-based programming aims to remove the programming complexity and make it accessible for end-users and non-engineers. To this end, a set of program blocks is designed that can be used to form various combinations and create complex programs. There are well-known block-based programming frameworks like Blockly [7], Scratch [10], [11], and Snap! [12].

Finally, text-based programming is a classic method for programming educational robots and requires minimal programming experience and knowledge. On the other hand, this method enables students to develop more complex projects.

Typically, hardware-based and visual programming methods are well-suited for younger learners due to their intuitive and hands-on approach. In contrast, block-based programming offers an accessible entry point for primary school students, providing a structured and visual environment for grasping programming concepts. Text-based programming, known for its more intricate syntax and logic, is better tailored for more advanced or mature students who possess a higher level of cognitive and conceptual understanding.

III. The proposed framework

This section introduces the proposed multi-player educational framework consisting of a Colorized Music-based Maze (CMM) based on piano notes and a set of simple robots trying to solve it. Here, piano is an example and can be replaced with various musical instruments.

In the designed framework, before the game begins, a CMM must be created and printed out as the playground. Sub-section III-A explains this process in detail. Next, robots must be prepared for the created CMM. Sub-section III-E describes the required robotic hardware and configurations based on a CMM. When a CMM is made, and robots are well configured, players should solve the designed maze by sending commands to their robots using piano notes. Robots will move or act appropriately based on the heard musical (i.e., received instructions). The workflow is detailed in Sub-section III-F.

A. Colorized Music-based Maze (CMM)

The generation of a Colorized Music-based Maze (CMM) is based on a piece of music, or pieces of music in the case of

a multi-player CMM. As a definition, a CMM is a maze that contains one or more start and end point pairs. Robots start at the start points and should find their path to related endpoints by following players' inputs.

To generate a CMM, first, single-octave music pieces are required. Each piece is a set of music notes in a specific order that can be translated into a list of piano keys, keeping the same order. This list will be assumed as a solution to a part of the CMM. The choice of single-octave pieces is due to two reasons: 1. There are a sufficient number of distinct notes in each octave, and 2. Different players can use different octaves to play simultaneously and without conflicts. Generally, there are three challenges in creating a CMM:

1) The number of possible robot movements on a square grid is limited to four; however, there are more piano notes in a single-octave music piece.
2) No matter how piano notes are assigned to actions, there can be cases where sequential notes would result in opposite actions that could cancel each other. For example, going forward being immediately followed by going backward. These notes could be omitted by players but would ruin the music.
3) By following music notes, there is no guarantee that a loop will not form. Players could omit the loop entirely, negatively affecting the music experience.

The rest of this section will address the mentioned challenges.

B. Challenge 1

Generally, in a piano, each octave comprises seven white keys and five black keys. Yet, in a square grid, there are only four possible movements for a robot. Therefore, in each music piece, four notes will be assigned to these movements. All other extra notes are called color locks. A color lock is a state for the robot so that the robot is locked and is waiting for a specific music note to unlock itself. In practical implementation, color locks can be specified on the CMM by particular color maze patches or walls. A robot can go into multiple sequential color locks based on the CMM.

To have a proper assignment of piano keys to instructions and color locks, firstly, notes translate to piano keys. Figure 1 depicts a music piece that contains this note-to-key translation.

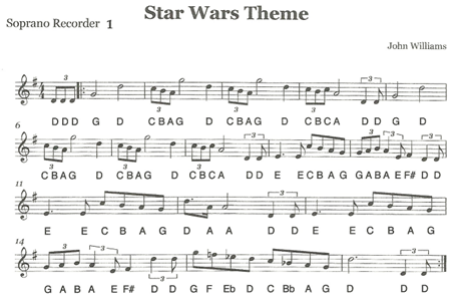

Fig. 1. A sample music piece and its translation to piano keys

Next, a list of all unique music notes in the music piece must be generated. This list is then required to be sorted in descending order by the number of appearances of each piano key in the music piece. Finally, movement instructions are assigned to the first four most repeated keys, and the remaining keys are assigned to color locks. Consequently, the generated CMM comprises minimum color locks and maximum movements. Table I shows the result of applying this procedure on the music piece in the example shown in Figure 1.

TABLE I
CMM INSTRUCTION-SET BASED THE MUSIC PIECE IN FIGURE 1

Note	Appearances	Assigned Action
D	25	t
A	17	b
G	13	l
C	12	r
B	11	Color Lock 1
E	8	Color Lock 2
F	3	Color Lock 3

C. Challenge 2

All CMM solutions must be parseable with the following grammar:

- $X \rightarrow T/B/L/R$
- $T \rightarrow T/L/R$
- $B \rightarrow B/L/R$
- $L \rightarrow L/T/B$
- $R \rightarrow R/T/B$

If a CMM solution fails to be parsed with the above-mentioned grammar, it has omittable instructions. Yet, music pieces do not have omittable parts. Figure 2 depicts the state machine of a robot solving a CMM using this grammar.

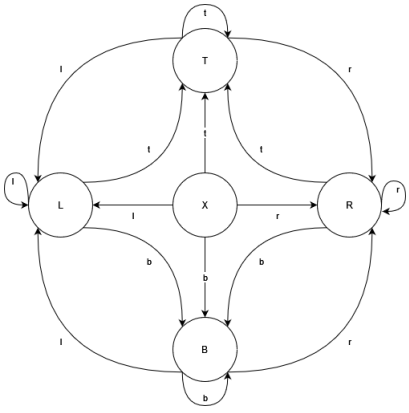

Fig. 2. state machine of a robot solving a CMM

To address the challenge of redundant instructions in a music piece, a supplementary state is added to the state machine in Figure 2 and is called the reverse state. The updated state machine is illustrated in Figure 3.

In generating a CMM, when there are two opposite instructions, the second instruction will move the robot to the reverse state. As it is shown in Figure 3, the robot will act

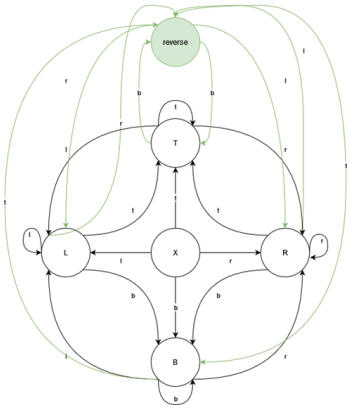

Fig. 3. Updated state machine of a robot solving a CMM

reversely from this state and the opposite instruction are no longer redundant, as are required to solve the maze.

It should be noted that the color for indicating reverse states must be different from the ones used for color locks. In practical implementation, reverse states must be specified on the CMM to be readable for robots.

D. Challenge 3

When creating a CMM solution based on a music piece, it is likely to generate routes that include loops. A loop is a redundant piece of the track from the start point to the endpoint and can be omitted. In contrast to direct opposite instructions, loops are too complicated to be solved by modifying the robots' state machine. To address this challenge, the following procedure (Algorithm III-D) is introduced:

1) While there are music notes in the music piece, create the CMM solution by following music notes until a loop is detected. If a loop is detected go to 2.
2) When a loop is detected, mark the current patch as a transition patch and start from an empty nearby patch. Go back to 1.
3) Connect transition patches.

In this algorithm, finding new start points can be done in multiple ways. Simple solutions are:

- Choose new start points in one specific direction (e.g., always choose a patch in an area right to the last generated part)
- Choose new start points on an Archimedean spiral (Figure 4).

Then, connecting the transition patches can be done using any basic path-planning algorithm. The only point is to keep the order of maze parts. A more complicated approach can be using a heuristic-based method to connect endpoints to start-points.

E. Robots' hardware requirements and configurations

This subsection will introduce the hardware requirements and features of the robots for solving CMMs. In general, any robot that has the following abilities can be used to solve CMMs:

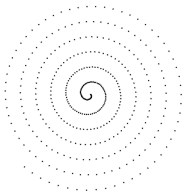

Fig. 4. Archimedean spiral

1) detects music notes.
2) moves appropriately on maze patches while not colliding with maze walls.
3) detects colors of maze patches or walls.
4) indicates different internal states (i.e., being in the reverse state or a color lock)

In an experimental study, as the most challenging part of the hardware design of CMM solver robots, the following minimal hardware was used to create a system to detect music notes:

- simple mic (LM358 as an amplifier)
- ATM328PU (Arduino UNO)

To detect different piano notes based on their frequencies, a Goertzel filter was used [13].

Experimental results showed that using the hardware mentioned above, the difference between two neighboring white piano keys' frequencies in the first three octaves is not sufficient or a reliable distinction, and they are not distinguishable. Thus, only octaves 4 to 7 can be utilized. This results in the fundamental limitation of using only four different robots in a CMM simultaneously. Otherwise, robots will be confused about which instructions to follow.

The reported experiment was demonstrative, and the mentioned limit can be removed by utilizing more powerful hardware for robots.

The only necessary configuration that robots need is to give them a map from colored patches (or walls) in the CMM to the corresponding color locks or the reverse state. Also, robots should be aware of automatic movement cells that connect CMM parts. Using these configurations, robots will be able to act correctly in the designed CMM.

F. Framework application workflow

This subsection introduces the workflow to use the proposed system. There are four stages in this system: creating a 1) CMM map, 2) implementing the real-world version of the CMM, 3) placing robots, and 4) playing the game. The rest of this sub-section describes these stages, respectively.

1) Creating a CMM: To produce a CMM, some pieces of music with the same or similar lengths should be chosen. It is possible to create a CMM with only one music piece, and then it will be a single-player CMM. As an example and for the rest of this chapter, the music piece in Figure 1 is selected.

After selecting the music piece(s), for each piece of music, the following steps should be done:

- Convert music notes to their corresponding piano keys. For the selected music, the output is *"DDDGDCBAGDCBAGDCBCADDGDCBAGD-CBAGDCBCADDEECBAGGABAEFDDEECBAG-DAADDEECBAGGABAEFDDGFDDCAAGDDD"*.
- The number of unique notes should be counted, and then movement keys and color locks should be assigned to the notes. Table I shows the output for our example. At this time, choose colors that can be printed and recognized by the robot for color-locks. It is recommended that selected colors be sufficiently distinguishable. If several color locks are placed in a row, a distinct color should be considered to represent them all.
- Using Algorithm III-D generate a maze for this piece of music.

Figure 5 shows a part of the generated CMM with out loops. Also, Figure 6 shows the connected CMM parts. In this figure, cells with a gray background are connections between small CMMs, and robots should follow them automatically (i.e., without user instruction).

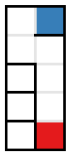

Fig. 5. a part of the generated CMM with out loops

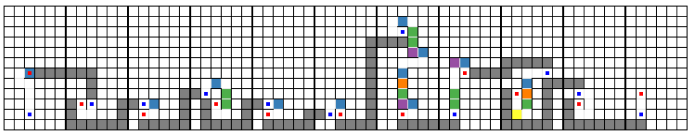

Fig. 6. the connected CMM parts using a basic method

2) Implementing the real-world version of the CMM: Several methods can be used to implement a CMM. A suggested method is to generate the CMM map utilizing a computer code and print it directly in the desired size. Another method is to use colored walls to make the CMM. This method is particularly suitable for small CMMs since it needs numerous colored walls.

3) Placing robots: In order to start the game, robots must be carefully placed at the starting points. Note that the starting points are different from each other, and each robot must be placed at its respective starting point. Otherwise, by following the path to the endpoint, the desired music will not be played.

4) Playing (solving a CMM): To solve a CMM, players must send movement commands to the robots or unlock robots in color lock situations with their pianos. It should be noted that a robot may go to multiple color lock states in the same CMM patch (e.i., without moving). Another important point is that in CMMs, there are patches where the robot will behave in reverse (i.e., reverse states).

IV. Conclusion

Compared to the reviewed educational robots, the proposed system has the significant advantage of providing a fun atmosphere for students to learn programming and music simultaneously. For young kids, playing with piano keys to send the correct instructions and solving the maze is challenging, but it will help them master the Piano by playing a game. On the other hand, older kids must improve their reasoning to create nice CMMs by selecting a suitable music piece and accurately following the procedure. Also, adjusting robot programs for solving different CMMs can be challenging in cases with multiple color-locks. Hence, tutors can use this system for students of a wide range of ages. Furthermore, since kids can play the designed games in groups, it is suitable to be used in a class and privately at home. If played in groups, the system allows students to collaborate in designing CMMs and competing to solve them. On the other hand, the required hardware is simple, so it keeps the final price of the system in a reasonable range. In conclusion, the proposed system is a suitable choice both for schools and private use.

References

[1] Miller, *Robotics for Education*. Cham: Springer International Publishing, 2016, pp. 2115–2134.

[2] O. Mubin, C. Stevens, S. Shahid, A. Mahmud, and J.-J. Dong, "A review of the applicability of robots in education," *Technology for Education and Learning*, vol. 1, 06 2013.

[3] F. Riedo, "Thymio: a holistic approach to designing accessible educational robots," *Thesis Number 6557, Federated Polytechnic School of Lausanne*, 01 2015.

[4] F. Mondada, M. Bonani, F. Riedo, Briod, P. Retornaz, and S. Magnenat, "Bringing robotics to formal education: The thymio open-source hardware robot," *IEEE Robotics & Automation Magazine*, vol. 24, no. 1, pp. 77–85, 2017.

[5] F. Riedo, M. Chevalier, S. Magnenat, and F. Mondada, "Thymio ii, a robot that grows wiser with children," in *2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, 2013, pp. 187–193.

[6] K. Mayerová, Z. Kubincová, and M. Veselovská, "Creating activities for after school robotic workshop with ozobot evo," in *2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2019, pp. 1–5.

[7] J. Trower and J. Gray, "Blockly language creation and applications," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, feb 2015.

[8] M. Jdeed, S. Zhevzhyk, F. Steinkellner, and W. Elmenreich, "Spiderino - a low-cost robot for swarm research and educational purposes," in *2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2017, pp. 35–39.

[9] M. Jdeed, M. Schranz, and W. Elmenreich, "A study using the low-cost swarm robotics platform spiderino in education," *Computers and Education Open*, vol. 1, p. 100017, dec 2020.

[10] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–15, nov 2010.

[11] J. Maloney, L. Burd, Y. Kafai, N. Rusk, and Silverman, "Scratch: a sneak preview [education]," in *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, 2004, pp. 104–109.

[12] K. Kahn, R. Megasari, E. Piantari, and E. Junaeti, "Ai programming by children using snap block programming in a developing country," in *Thirteenth European Conference on Technology Enhanced Learning*, vol. 11082. Springer, 2018.

[13] R. Beck, A. Dempster, and I. Kale, "Finite-precision goertzel filters used for signal tone detection," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 7, pp. 691–700, 2001.