2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #40048

A Systematic Approach to Teaching the Foundational Concepts of Programming Using LEGO and Matlab

Il Yoon, University of North Georgia

A Systematic Approach to Teaching the Foundational Concepts of Programming Using LEGO and Matlab

Il Yoon

University of North Georgia

Georgia, USA

iyoon@ung.edu

Abstract

Learning programming can be challenging for newly admitted engineering students. It may be even more challenging in two-year colleges due to the broad spectrum of student populations, which includes both traditional and non-traditional students. Students who have been out of school for several years after high school may find programming particularly challenging, while students who have just graduated from high school may find it less so. To overcome this challenge, it is suggested to teach programming using familiar tools from the very beginner level to the intermediate level. This paper proposes a new course to teach the foundational concepts of programming using LEGO EV3, a robotics toolkit, and MATLAB to control the LEGO robots. The course consists of three parts: Basics, Assigned Projects, and Student-Led Projects. In the Basics part, students learn fundamental programming concepts such as algorithms, flowcharts, variables, input/output, IF statements, and loop structures. Systematically designed application problems are provided to enhance understanding of programming basics. In the Basics part, students write code to control LEGO robots and solve application problems. This helps students overcome their unfamiliarity with programming, as they can see how the robots move interactively as they program. After completing the Basics part, students will work on two assigned projects: a line-tracking robot and a cleaning robot. Guided problems are provided to help students complete these projects. After students complete the assigned projects, students will undertake a Student-Led project. For this project, they will design an engineering product of their choice and build it using LEGO EV3 and MATLAB. To evaluate the effectiveness of the new course, the level of difficulty and repetition of application problems and guided problems were measured, along with the applicability of active learning activities. The measured data shows that the level of difficulty gradually increases overall from the beginner level, which helps students build foundational concepts of programming as beginners. It also shows that the topics in the Basics part are systematically repeated in the application problems and guided problems, which enhances learning retention of the topics. Additionally, the data shows that the learning methods used in this course lead to a higher percentage of active learning activities with higher complexity, making this course more beneficial for students. To evaluate the effectiveness of this proposed course, it is recommended to offer this course for a few semesters and conduct surveys at the end of each semester to collect feedback and measure its effectiveness.

1 Introduction

Programming is one of the most important skills that engineering students need to learn. However, it is known that learning programming is not easy for students. A study has shown that many college-level students still struggle with programming even after taking an introductory programming course [1]. This means that teaching programming is very challenging. Teaching programming at two-year colleges is even more challenging due to the broad spectrum of student populations. Some students may have just graduated from high school, while others may have been out of school for several years after graduating. If these students have not been involved in STEM areas during their time out of school, it can be very difficult for them to learn programming due to unfamiliarity. This makes it very challenging for instructors to prepare the curriculum for a programming course due to the various levels of background knowledge of students in the class. Therefore, there is a need to develop a course that helps students build a solid foundational understanding of programming using familiar tools, starting from the beginner level and progressing to the intermediate level.

One effective way to improve understanding in the learning process is through hands-on experiences. In this newly suggested course, LEGO EV3 and MATLAB are introduced to improve the perceptibility of learning programming as a tool for hands-on learning. In the new course, students will simultaneously observe how their code works as they control a LEGO robot by writing code.

Although LEGO robotic products have already been used in college-level courses for learning programming, this newly suggested course offers some advantages for effective programming learning. Firstly, this new course uses MATLAB to control LEGO robots. Previous programming languages used for learning programming using LEGO robots were NQC [2], [3], NXC [3], ROBOLAB [2], [4], Legos Mindstorm iconic language [5], Java [6], and Visual Basic [2]. Out of these options, MATLAB was chosen for this new course for several reasons. First, it is one of the essential programming languages that many engineering students are required to learn. Second, it provides clear and intuitive syntax, enabling students to transform their algorithms into programming texts quickly [7]. If a difficult and complicated language is used, students would struggle with the language itself, rather than trying to understand the concept of programming. Thus, using MATLAB helps students focus more on understanding the concept of programming. Lastly, MATLAB is a text-based language. Although graphical programming languages are more accessible for beginners in general, it is essential for engineering students to be familiar with a text-based programming language as professionals in their future [8].

This new course focuses on the immediate and interactive application of learning by providing systematically designed application problems. After students learn a new programming concept, they can apply it instantly by exercising related application problems using LEGO robots. As students work on application problems, they can interactively observe how LEGO robots are controlled as they program.

Using MATLAB and LEGO has been introduced previously for engineering education [9], [10]. However, their works did not necessarily focus on learning the foundational concepts of programming. Therefore, it is worth developing a new course specifically to help engineering students learn and practice these foundational concepts using LEGO and MATLAB. Accordingly, a new course using LEGO EV3 and MATLAB was developed and presented in this paper. Although the new course was carefully designed with a systematic approach, it is necessary to ensure its effectiveness before offering it. To validate the effectiveness of the new course, the level of difficulty, repetition, and applicability of active learning activities were measured, and the results are presented.

2 Setup descriptions

In this newly proposed course, LEGO Mindstorms EV3 is used with the MATLAB Support Package provided by MathWorks to control the LEGO robot using MATLAB. The sensors used in this course are: one color sensor, one ultrasonic sensor, two touch sensors, and one gyroscope sensor. The motors used are two large motors and one medium motor, with the two large motors being identical to each other. There are three methods of connection available between a user's computer and the LEGO controller: USB cable connection, Bluetooth connection, and Wi-Fi connection. In this new course, the LEGO robot is connected to a computer via USB connection for better connection stability.

3 Course descriptions

The newly proposed course is composed of two main parts: the Basics part and the Projects part, both of which utilize LEGO and MATLAB. In the Basics part, students learn and practice fundamental programming concepts. The Projects part consists of two assigned projects, a line-tracking robot and a cleaning robot, and one Student-Led project. Through the assigned projects, students apply the basic programming knowledge they learned in the Basics part to engineering applications. The Student-Led project allows students to solidify their programming knowledge by using MATLAB to build an engineering product of their choice. Systematically designed application problems and guided problems are provided to help students understand programming concepts at each step of learning in the Basics part and the two assigned projects in the Projects part.

This proposed course is designed for 1 – 2 credit hours with a recommended minimum of 2 contact hours per week to provide enough time for students to practice in class. This course is suggested to be offered as a full college-level course. However, the curriculum of this proposed course can be integrated into an existing introductory engineering course. At the author's institution, the curriculum was successfully integrated into the 'Introduction to Engineering' course for two semesters from Fall 2017 to Spring 2018. The course's main objective was to provide students with general knowledge of engineering. However, the curriculum also included basic programming skills as part of the learning objectives.

3.1 Basics part

The Basics part consists of sections on variables, input/output, IF statements, and loop structures. From this part, students learn and practice the very basics of programming. Firstly, students learn commands and syntax, and how to use them to write code from an instructor. Then, application problems are given so that students gradually build up their programming knowledge through problem-based learning, which is known to be effective [11], [12]. Each application problem is given right after a new concept is taught at each section. Students write code using MATLAB to control the LEGO as directed in the application problems. This helps students understand how their code works by writing the code and watching how it controls the LEGO simultaneously. The LEGO product is used to practice input/output, IF statements, and loop structures, while it is not used for the exercise of variables. This is because it is not necessarily more effective to understand the concept of variables using LEGO, as learning variables is relatively simple compared to other topics. For this reason, describing how to practice variables is skipped in this paper because the main purpose of this paper is to describe how effective it is to use LEGO and MATLAB to learn programming. Application problems are designed cumulatively so that students practice new concepts and review previous concepts simultaneously when they move to a new application problem from the previous one.

3.1.1 Input/Output

In the section on input/output, students learn how to assign values to variables using INPUT command, as well as how to output data stored in a variable using the DISP command. After learning how to use these commands, students exercise them with a LEGO car. They are asked to build a simple toy car using the LEGO product and write code that includes INPUT and DISP commands to control the car to move forward and backward at various speeds, as

directed in the application problems. The first and last application problems, along with the expected Matlab codes for the input/output section, are shown below.

- Determine the speed and direction of the LEGO car. Then, use INPUT command to display "**Input the duration of time to move forward"**, and assign any input value into variable **time_frwd**. Then, make the LEGO car move forward for the time input.

```
motor_1.Speed = 50;
time_frwd = input('Input the duration of time to move forward ');
start(motor_1);
pause(time_frwd);
stop(motor_1);
```

- Use INPUT command to display "**Input the duration of time to move backward."** and assign any inpt value into variable **time_bkwd**. Use INPUT command to display **"Input the speed of the car and direction."** and assign any value into variable **speed_bkwd**. Then, make the car move backward for the time duration input and with the speed input. Then, use INPUT command to display "**The duration of time for moving backward is"**. Use INPUT command to display the time duration that the car move backward. Use INPUT command to display "**The speed for moving backward is"**. Use INPUT command to display the speed when the car move backward.

```
time_bkwd =input("Input time to move in seconds. ");
speed_bkwd =input("Input speed to move forward. ");
motor_1.Speed = speed_bkwd;
start(motor_1);
pause(time_frwd);
stop(motor_1);
disp('The duration of time for moving backward is ');
disp(time_bkwd);
disp('The speed for moving backward is ');
disp(speed_bkwd);
```

3.1.2 IF statements

In the section on IF statements, students learn how to use IF, ELSE, and ELSEIF commands. After learning these, they are asked to build a LEGO car with a color sensor connected. Then, students write codes as directed in the application problems to make the LEGO car move forward, backward or stop depending on the detected colors on the floor using the color sensor. While students solve the application problems, they are also asked to use input and DISP commands as directed in the problems to review the section on input/output. The following shows the first and last application problems and the corresponding MATLAB codes for the section on IF statements.

- Detect a color using a color sensor and display **"Green"** if the detected color is green.

```
color = readColor(mycolorsensor);
if strcmp(color,'green')
   disp('Green');
```

```
end
```

- Use INPUT command to display "**Input an wait time to start**." and assign the value to variable **time_start**. Detect a color using a color sensor. If the color is green, make the car move forward immediately. If the color is red, make the motor move forward after the time stored in the variable **time_start**. Detect a color using the color sensor again. If the detected color is green, display "Keep moving forward.". If the detected color is blue, make the car move 2 times faster. If the color is red, make the car stop after 1 second.

```
time_start = input('Input an wait time to start. ');

color = readColor(mycolorsensor);
if strcmp(color,'green')
   motor_1.Speed = 10;
   start(motor_1);
elseif strcmp(color,'red')
   pause(time_start);
   motor_1.Speed = 10;
   start(motor_1);
end

color = readColor(mycolorsensor);
if strcmp(color,'green')
   disp('')
elseif strcmp(color,'blue')
   motor_1.Speed = 20;
elseif strcmp(color,'red')
   pause(1);
   stop(motor_1);
end
```

3.1.3 Loop structures

In programming, understanding loop structures is important but can be difficult for beginners. Therefore, it is crucial for beginners to have a solid understanding of how to use loop structures when learning. In this course, students first learn the basics of FOR and WHILE loops. Then, they assemble a LEGO car with a color sensor, an ultrasonic sensor, a gyro sensor, two motors, and a controller. Students then write code as directed in the application problems to make the car move forward and backward repeatedly, or rotate depending on detected colors, distance from the wall, or angle that the car rotates using FOR and WHILE loops. While solving the application problems, students are also asked to use INPUT, DISP, and IF statements as directed in the problems for review purposes. The following shows the first and the last application problems and the corresponding Matlab codes for the section of loop structures.

- Make the car move forward for 2 seconds, and stop. Then, make the car move backward for 2 seconds. Repeat this 3 times using a FOR loop. The speed parameter is 10.

```
for i=1:3
```

```
    motor_1.Speed = 10;
    start(motor_1);
    pause(2);
    stop(motor_1);

    motor_1.Speed = -10;
    start(motor_1);
    pause(2);
    stop(motor_1);
end
```

- Refactor the code from problem 3 to reduce the duplicated lines.

```
spd = -10;
count_repeat = 0;
while count_repeat<3
    for j=1:2
        spd = -spd;
        motor_1.Speed = spd;
        start(motor_1);
        pause(2);
        stop(motor_1);
    end
    count_repeat = count_repeat + 1;
end
```

3.2 Projects part

After students learn basic programming concepts from the Basics part, two assigned projects and one Student-Led project are given to them to enhance their understanding by applying their basic programming knowledge to engineering problems. They are asked to work on the projects as a team because teamwork is one of the necessary skills that employers want [13]. The number of students per team is 2 to 4. Students need to integrate all learned programming knowledge of variables, input/output, IF statements and loop structures, and utilize them to complete the projects. The two assigned projects are making a line tracking robot and a cleaning robot. For the line tracking robot, students build a LEGO car and write code to make the car follow a line on the floor. For the cleaning robot, students build a LEGO robot with a cleaning tool attached and write code to make the robot move around while it picks small pieces of papers on the floor. For these two assigned projects, guided problems are given to students to help them complete their projects. Though students learn the basics of programming from the Basics part, it is not easy for them to complete the assigned projects on their own because they are not familiar with making somewhat complicated codes for their projects yet. Thus, the guided problems for each assigned project guide students step-by-step to help complete the projects. For the Student-Led project, students determine an engineering product that they want to develop. Then, they design and build the product using LEGO and write code to control. Some accessories are allowed to be used when building the engineering product. For the Student-Led project, no guided problem is given. That is, students are asked to make their code on their own entirely.

4 Effectiveness of the Course

To validate the effectiveness of the new course, three categories were measured: the level of difficulty, repetition, and applicability of active learning activities. This was done to ensure that students were able to understand the foundational concepts, apply their knowledge, recall what they had learned, engage actively in class work, and experience the effectiveness of using LEGO while learning programming.

4.1 Level of difficulty

An important factor considered when designing this course was the application of knowledge. It is expected that students can have a better understanding of programming when they apply their programming knowledge to tangible applications. Hence, application problems and guided problems were designed to help students understand better by controlling the LEGO robot using MATLAB. It is important that students start exercising problems from the beginner level, with the difficulty of the codes gradually increasing. To ensure this, the level of difficulty of each application problem and guided problem was measured and provided in this section..

To assess the level of difficulty of each problem, the author created expected answer codes for each problem, and a method modified from a previous study [14] was used. The following items were used to measure the level of difficulty of the application and guided problems in this course: the number of commands, the number of operators, cyclomatic complexity, the maximum nested blocks, and the number of reduced lines after refactoring. Each answer code was measured and scored based on these items. The scores of all items for each problem were then added to predict its level of difficulty.

Fig. 1 shows that the level of difficulty of input/output application problems increases overall as the problem number increases due to the increase in the number of commands. For IF statements application problems as the next topic in Fig. 2, the level of difficulty increases gradually overall. The level of difficulty increases mainly due to the increase in the number of commands. The cyclomatic complexity overall increases a little bit due to the increase in the number of commands. In the loop structures application problems as the next topic in Fig. 3, the level of difficulty increases overall though there are some fluctuations. The maximum nesting depth of the application problems is 1, except at problem 5, problem 13, and problem 14. The reduced lines after refactoring are 4 at problem 13 and problem 14 while they are 0 at other problems. In the line tracking robot project section, the level of difficulty of the guided problems increases gradually as shown in Fig. 4. Compared to the level of difficulty in Fig. 1 to Fig. 3, it shows that the level of difficulty of the last problem is much higher than the one of the first problem in Fig. 4. All factors except reduced lines after refactoring show gradual increases though the rates of increases are varying. The level of difficulty of the cleaning robot guided problems is shown in Fig. 5. It shows gradual increases from problem 1 to problem 5, a sudden increase from problem 5 to problem 6 and a decrease from problem 6 to problem 7. There is a sudden increase in reduced lines after refactoring from problem 6 to problem 7, while the number of commands, number of operators, and cyclomatic complexity decrease. In Fig. 6, the level of difficulty of each problem for all sections is shown. This displays how the level of difficulty of problems through all sections increases as well as in each section. This graph exhibits that the level of difficulty is mostly under 20 for the sections of input/output, IF statements and loop structures. For the guided problems of the

line tracking robot and cleaning robot, the level of difficulty increases from under 20 to 63 for the line tracking robot, and to 104 for the cleaning robot.

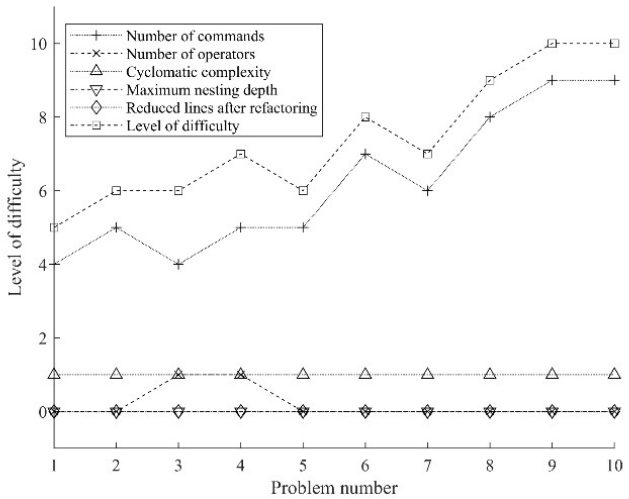

Fig. 1. Level of difficulty of input/output application problems

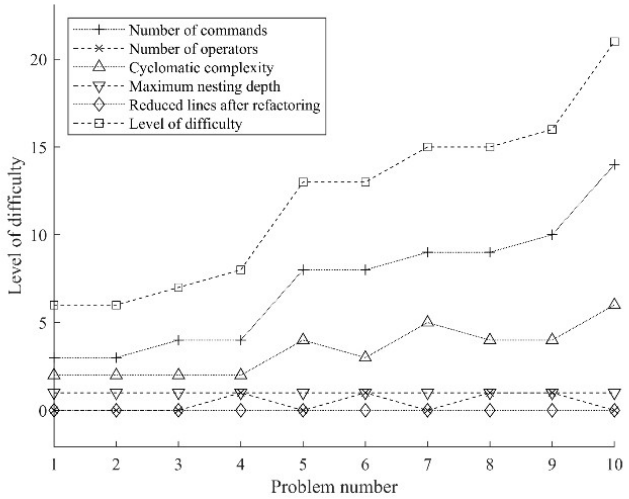

Fig. 2. Level of difficulty of IF statements application problems

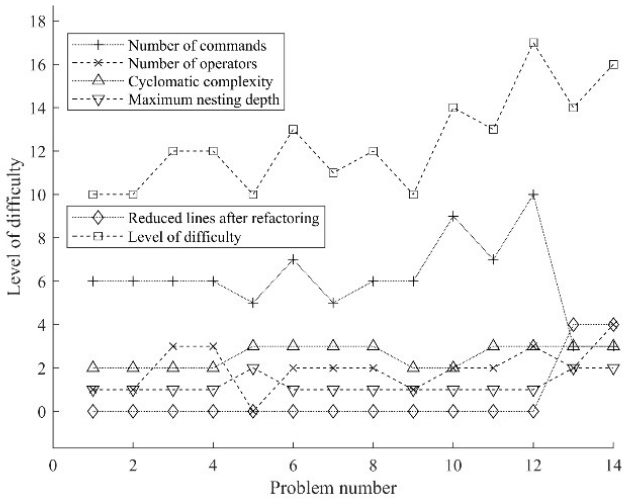

Fig. 3. Level of difficulty of loop structures application problems

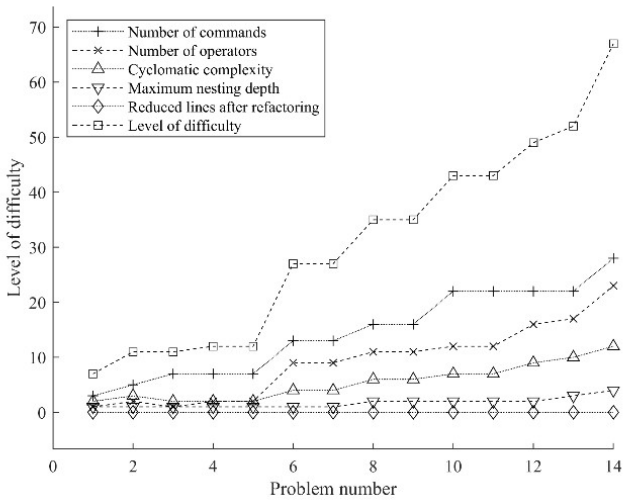

Fig. 4. Level of difficulty of the line tracking robot guided problems

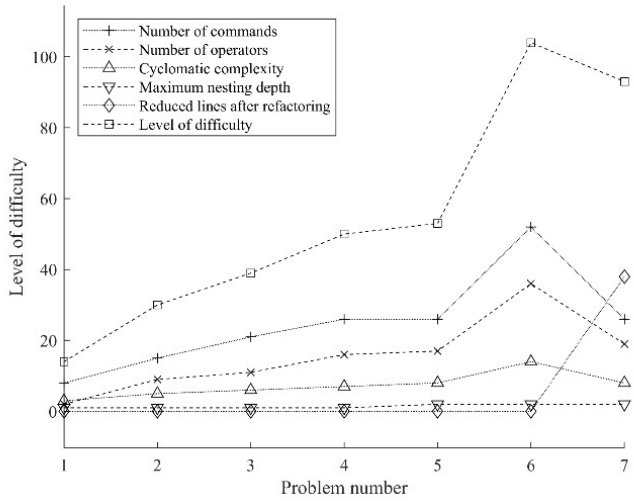

Fig. 5. Level of difficulty of the cleaning robot guided problems

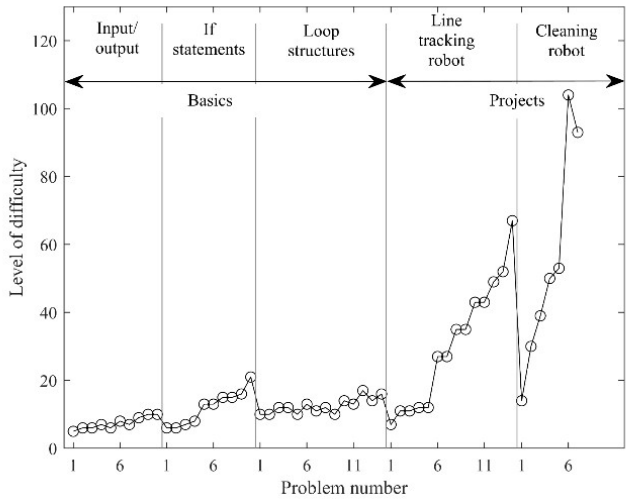

Fig. 6. Level of difficulty of each problem for all sections

4.2 Repetition

In Table 1, it shows the number of repetition of input/output commands, IF statements and loop structures in the application problems of each section. According to the table, students are asked to use input/output commands 15 times while they exercise the application problems of the section of IF statements, 24 times while they exercise the application problems of loop structures, 5 times while they exercise the guided problems of the section of line tracking robot, and 6 times while they practice the guided problems of the section of the cleaning robot. Similarly, students are asked to use IF statements 4 times while they practice the application problems of the section of loop structures, 23 times while they practice the guided problems of the line tracking robot, and 3 times while they practice the guided problems of the cleaning robot. In a similar way, students are asked to use loop structures, either a FOR or WHILE loop, 29 times while they practice the guided problems of the section of the line tracking robot, and 29 times while they practice the application guided of the

section of the cleaning robot. Eventually, input/output practice is repeated 50 times, if statement practice is repeated 30 times and loop structure practice is repeated 58 times. In this way, students repeat what they learn so that they can recall easily.

Table 1. Number of repetitions of each topic in each section

	Section					Total
	Input/ output	IF	Loop	Line tracking	Cleaning	
Input/output	NA	15	24	5	6	50
IF	NA	NA	4	23	3	30
Loop	NA	NA	NA	29	29	58

4.3 Active learning

Performing projects using LEGO and MATLAB in this newly suggested course can be considered as active learning activities. Then, it would be questioned how much it would be effective as active learning for learning programming. A study suggested 22 activities in total as active learning activities [15]. The 22 activities are categorized into three levels based on complexity: low complexity, moderate complexity and high complexity. In Table 2, it shows the applicability of the 22 active learning activities to conducting projects using LEGO and MATLAB in this course. In the table, conducting projects using the LEGO and MATLAB suggested in this course can be considered as active learning activities described in code B, D, G and K in the low complexity, code L, P and Q in the moderate complexity, and code S, T and V in the high complexity. This shows that 4 activities out of 11 activities in the low complexity can be considered as active learning activities in the new course, which is 36 % out of 100 %. In the moderate complexity, 3 activities out of 7 activities, which is 43 % out of 100 %, can be considered as active learning activities. In the high complexity, 3 activities out of 4 activities, which is 75 % out of 100 %, can be considered as active learning activities. Consequently, the percentage increases as the complexity becomes higher.

Table 2. Applicability of active learning activities to the newly suggested course [15]. The codes with A are applicable and NA are not applicable to this new course.

Low Complexity										
A	B	C	D	E	F	G	H	I	J	K
NA	A	NA	A	NA	NA	A	NA	NA	NA	A
Moderate Complexity							High Complexity			
L	M	N	O	P	Q	R	S	T	U	V
A	NA	NA	NA	A	A	NA	A	A	NA	A

5 Discussions

The assessment of effectiveness shows that this course was designed systematically to enhance students' understanding of programming. The level of difficulty of application problems and guided problems overall increases in each section, as shown from Fig. 1 to Fig. 5. This helps students build the concept of input/output, IF statements, and loop structures

gradually. Especially for the section of input/output and IF statements, the application problems were designed to be simple, so students can focus on learning these concepts. For example, in Fig. 1, the application problems in the input/output section have only 2 operators, and the maximum nesting depths and reduced lines after refactoring are 0. This means the application problems in this section are quite simple, while the difficulty level increases gradually. This approach also applies to the application problems for IF statements in Fig. 2, where the level of difficulty increases gradually mainly due to the gradual increase in the number of commands, so students can focus on understanding IF statements. Because learning programming is likely new for most students, this gradual increase in difficulty would be helpful for them to become familiar with programming and understand the concepts.

Loop structures are often considered challenging for new learners in programming. Therefore, it is recommended that the application problems for loop structures should not be too difficult so that students can fully understand the concept of loop structures. In Fig. 3, the maximum nesting depth of the application problems is 1, except for problems 5, 13, and 14. This means that the application problems were designed to focus on the simple application of loop structures without considering complicated nesting. This approach will help students build a solid understanding of loop structures. One noteworthy aspect of this section is the introduction of refactoring in problems 13 and 14. Learning and understanding refactoring can help students make their code more efficient..

In the section of the line tracking robot project shown in Fig. 4, the level of difficulty of the last problem is much higher than the first problem. It is expected that students are not yet familiar with transforming a complicated algorithm such as the tracking robot project into code. In particular, the level of difficulty of the expected completed code for this project is close to 70 while the maximum level of difficulty in the Basics part is below 25. That means it would be very difficult for students to start working on the projects with only exercising problems at the difficulty level below 25 in the Basics part. Therefore, starting with a guided problem under the difficulty level of 25 in the Projects part would help students experience a smooth transition from the Basics part to the Projects part to apply their knowledge learned from the Basics part to the line tracking robot project and the cleaning robot project. Another thing worth mentioning is that the maximum nesting depth increases from 1 to 4 in this section. By experiencing the maximum nesting depth of 4, students are introduced to nesting for more complicated codes for future projects.

The number of repetitions in Table 1 shows that the new course was designed systematically to repeat what students have learned, enhancing their retention. However, it may be thought that the number of repetitions for IF statements, which is only repeated four times in the section of loop structures, is not enough. Nevertheless, this may help students focus on learning loop structures without the added complexity of IF statements, which could be confusing for beginners. Thus, minimizing the repetition of IF statements in the loop structures section could be a useful approach for students to focus on loop structures. It is also worth mentioning that the application problems for repetition are not simply recalling what students learned; rather, they require students to apply what they recall to given problems, thereby strengthening their understanding of programming concepts while they repeat.

As mentioned in the Results section, Table 2 shows that the percentage of active learning activities increases as the complexity of the activities becomes higher. This suggests that

using LEGO and MATLAB to conduct projects in this new course is more suitable for higher complexity active learning activities. Although it is not explicitly stated in the cited study [15], it can be assumed that activities with higher complexity are more effective as active learning activities because they require and draw students' attention and participation more deeply into the activity. Given that this new course has a higher percentage of higher complexity activities, it can be concluded that conducting projects using LEGO and MATLAB as designed in this newly suggested course would be effective for students to have better learning experiences in programming due to the effects of active learning. However, this new course can be further improved by adding more low complexity active learning activities without much additional preparation, as instructors can add them with relative ease due to their simplicity. One thing worth to mention is that EV3 is no longer available, so new edition of LEGO robotics can be used.

6 Conclusions

The proposed new course aims to teach the foundational concepts of programming to engineering students using LEGO EV3 and MATLAB. The effectiveness of the course was measured by assessing the level of difficulty, repetition, and applicability of active learning activities. The results indicate that the level of difficulty increases gradually in each section, and the foundational concepts of input/output, IF statements, and loop structures are systematically repeated throughout the course. The level of difficulties of the line tracking robot and the cleaning robot projects increase gradually from the Basics section. The measurement of active learning shows that the course is suitable for higher complexity active learning activities. Overall, the proposed course is promising in effectively teaching programming to engineering students. it is recommended to offer this new course for several semesters and conduct surveys to collect feedback and measure its effectiveness.

7 References

[1] M. McCracken *et al.*, "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," *ACM SIGCSE Bull.*, vol. 33, no. 4, p. 125, Dec. 2001, doi: 10.1145/572139.572181.

[2] M. A. Garcia and H. P. Mc-Neill, "Learning how to develop software using the toy LEGO mindstorms," in *Proceedings of the 7th annual conference on Innovation and technology in computer science education - ITiCSE'02*, Aarhus, Denmark, 2002, p. 239. doi: 10.1145/544414.544508.

[3] A. Pásztor, R. Pap-Szigeti, and E. Török, "Mobile Robots in Teaching Programming for IT Engineers and its Effects," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 11, 2013, doi: 10.14569/IJACSA.2013.041123.

[4] E. Wang, J. LaCombe, and A.-M. Vollstedt, "Teaching Structured Programming Using Lego Programmable Bricks," Jun. 2007. [Online]. Available: https://peer.asee.org/2160

[5] N. E. Naga, S. J. Stokes, and H. E. Naga, "The Use Of Real Time Operating Systems And Development Tools As A Mean To Revitalize Computer Engineering Programming Courses," Salt Lake City, Utah, Jun. 2004. [Online]. Available: https://peer.asee.org/13110

[6] D. J. Barnes, "Teaching introductory Java through LEGO MINDSTORMS models," in *Proceedings of the 33rd SIGCSE technical symposium on Computer science education - SIGCSE '02*, Cincinnati, Kentucky, 2002, p. 147. doi: 10.1145/563340.563397.

[7] H. Fangohr, "A Comparison of C, MATLAB, and Python as Teaching Languages in Engineering," in *Computational Science - ICCS 2004*, vol. 3039, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1210–1217. doi: 10.1007/978-3-540-25944-2_157.
[8] D. Weintrop and U. Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–25, Oct. 2017, doi: 10.1145/3089799.
[9] A. Behrens *et al.*, "MATLAB Meets LEGO Mindstorms—A Freshman Introduction Course Into Practical Engineering," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 306–317, May 2010, doi: 10.1109/TE.2009.2017272.
[10] M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino, "A LEGO Mindstorms multi-robot setup in the Automatic Control Telelab," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 9812–9817, Jan. 2011, doi: 10.3182/20110828-6-IT-1002.02080.
[11] D. Budny, L. Lund, and R. Khanna, "Designing Service Learning Projects for Freshman Engineering Students," *Int. J. Eng. Pedagogy IJEP*, vol. 3, no. S1, p. 31, Feb. 2013, doi: 10.3991/ijep.v3iS1.2371.
[12] M. A. Albanese and S. Mitchell, "Problem-based learning: a review of literature on its outcomes and implementation issues [published erratum appears in Acad Med 1993 Aug;68(8):615]," *Acad. Med.*, vol. 68, no. 1, pp. 52–81, Jan. 1993, doi: 10.1097/00001888-199301000-00012.
[13] G. M. Lundberg, B. R. Krogstie, and J. Krogstie, "From Employable to Fully Operational: The Need for Training of Computer Science Graduates," *Int. J. Eng. Pedagogy IJEP*, vol. 11, no. 3, p. 21, May 2021, doi: 10.3991/ijep.v11i3.19995.
[14] S. Elnaffar, "Using software metrics to predict the difficulty of code writing questions," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, Abu Dhabi, United Arab Emirates, Apr. 2016, pp. 513–518. doi: 10.1109/EDUCON.2016.7474601.
[15] J. A. Van Amburgh, J. W. Devlin, J. L. Kirwin, and D. M. Qualters, "A tool for measuring active learning in the classroom," *Am. J. Pharm. Educ.*, vol. 71, no. 5, p. 85, Oct. 2007.