

A Toolkit to Learn Algorithmic Thinking using mBot Robot

Irnayanti Dwi Kusuma^{†1} Fitri Utamingrum^{†2}
Tetsuro Kakeshita^{†3}

Abstract: We propose a new toolkit to help children at elementary or secondary school to learn algorithmic thinking through mBot robot programming in this paper. The toolkit also helps people teaching children such as teachers and parents. The toolkit is composed of four types of related elements: specification, mBot hardware component, algorithm, and Scratch code. A toolkit user can understand the robot's behavior through the algorithm and specification even if they do not know Scratch programming language and can convert each step of an algorithm to a sequence of Scratch code. The toolkit also provides a library for the users to create a new mBot application and associated algorithm and code.

Keywords: mBot Robot, Programming Toolkit, Educational Robotics, Algorithmic Thinking

1. Introduction

Nowadays, new technology, such as robotic kit and computer programming, is growing fast. Many children play with technological toys during their playing time. Some companies develop robots which can be used even for infants to introduce and educate robotics programming. Countries such as Japan, UK, EU and US have programming curriculum for elementary and secondary school. Almost all schools use the products of such robotic companies to teach programming easily and to take the children play with fun. On the other hand, teachers and parents who bought the robotics kit often do not have enough knowledge about algorithmic thinking [2,7]. However, by mastering algorithmic thinking, students can systematically solve problem and can implement new ideas more easily.

During elementary and secondary school age, children have a creative and fluid mindset that allows them to think in a more out of the box way. Thus, we can utilize their growth period to extend their ability and to set their mindset with algorithmic thinking. It is important to understand algorithm before writing a scratch code because it will facilitate abstract level thinking and creativity. In this paper, our goal is to teach algorithmic thinking through robot programming. We propose a programming toolkit for this purpose. The toolkit is composed of four types of elements: specification, mBot hardware component, algorithm and Scratch code. These components are related each other so that a user can easily understand the Scratch code using the corresponding algorithm and specification. A toolkit user can also utilize the specifications and the algorithms to create new ideas. All such activities are useful to learn algorithmic thinking.

We develop the toolkit by utilizing a product called mBot (Figure 1) by MIT Media Lab Lifelong Kindergarten. The product is composed of a mBot robot to run a program and mBlock IDE to write scratch code. In this toolkit, we provide a library useful for people who accompany children that do not have enough knowledge about algorithmic thinking. The toolkit provides a rich set of sample programs that covers all

functions of mBot hardware components to solve a problem utilizing algorithmic thinking.



Figure 1 The mBot Robot

This paper is organized as follows. We introduce hardware and the associated software development environment for mBot robot in Section 2. Sections 3 and 4 are the heart of this paper. After explaining the overview of the proposed toolkit in Section 3, we shall explain 6 programs contained in the toolkit in detail. Section 5 contains preliminary evaluation of the toolkit. We shall explain the difference of our contribution compared to the related work in Section 6. Finally, we provide concluding remarks and future work in Section 7.

2. Overview of mBot Robot

2.1 Hardware Organization

mCore is the main board which has many parts of an electronic module of mBot (Figure 2). A mCore initially has the following eight electronic modules.

1. RJ25 Port: Connect additional sensors
2. RGB LED: Output light of the specified color defined using RGB code
3. Buzzer: Output buzzer tone defined by frequency
4. IR Receiver: Receive information from remote controller
5. Light Sensor: Generate value by sensing the brightness of ambient light
6. IR Transmitting: Send information composed of numbers and text
7. Button: Get value of the button
8. Motor port: Control motor of the mBot. Two motor ports are equipped to independently control left and right wheels.

Besides the above modules, we can also use additional modules, such as ultrasonic sensor to detect object, line patrolling sensor to detect black line, and Bluetooth module to enjoy wireless programming.

^{†1,2} Brawijaya University, Indonesia
^{†3} Saga University, Japan

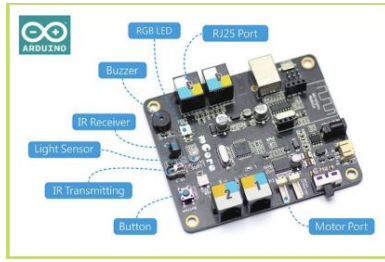


Figure 2 mCore Main Board

2.2 Programming Environment

mBlock provides programming environment for mBot robot. mBlock is a software to control mCore main board and achieve the corresponding functions. mBlock can be downloaded from <http://mblock.cc/download>. Main Interface of mBlock is illustrated in Figure 4. It will automatically convert the scratch code (as in figure 6) to Arduino code. Therefore, mBlock is an IDE to write scratch code and mBot is a robot, which can run the scratch code in online or offline mode (Arduino based).

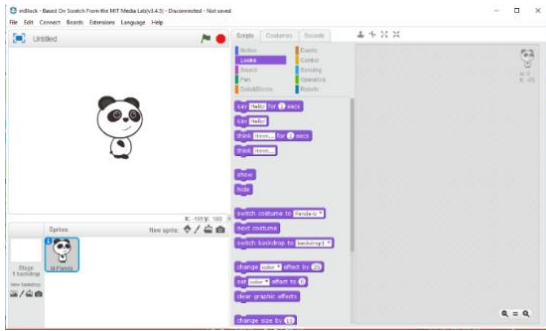


Figure 3 Main interface of mBlock

An mBot can be connected to mBlock using USB cable, Bluetooth or 2.4G. After connecting mBlock and mBot, a user can enable communication using mBlock by selecting an appropriate COM port as illustrated in Figure 5. mBot can be controlled by mBlock only when the port is connected [3,8].

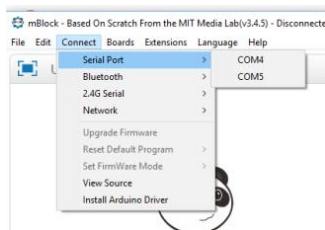


Figure 4 Connecting mBlock to mBot

Furthermore, mBot is an Arduino based robots, where Arduino is an open-source electronics platform based on easy-to-use hardware and software [1]. mBot can run with offline mode by uploading a program to the main board by the following steps:

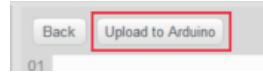
1. Allow program code to be uploaded to mBot’s main controller and make the robot operate offline



2. Then right-click, select ‘Upload Arduino program’ as shown below



3. Click the ‘Upload to Arduino’ in the opened window



When uploading the scratch code to Arduino, make sure that your USB cable is connected to mBot’s main controller and the corresponding port number is correct.

3. Overview of Programming Toolkit

3.1 Toolkit Organization

The toolkit is composed of four types of elements: specifications, mBot hardware component, algorithms and scratch code (Figure 5). A specification describes a purpose or a story of a program to control mBot robot. mBot hardware components must be programmed to realizes the specification. An algorithm describes a sequence of steps of basic operations, conditional statements, iterative statements and subroutine calls which describes instructions to mBot. An algorithm step is described in a natural language and each basic operation corresponds to a mBot hardware component. Thus, there is a mapping between mBot hardware component and basic operations of algorithms. On the other hand, each algorithm step corresponds to a sequence of scratch code implementing the step. Note that each step of a scratch code corresponds to an algorithm step. As a result, there are three types of mapping within the toolkit: the mapping between specification and algorithm, the mapping between mBot hardware component and algorithm step, and the mapping between algorithm step and scratch code.

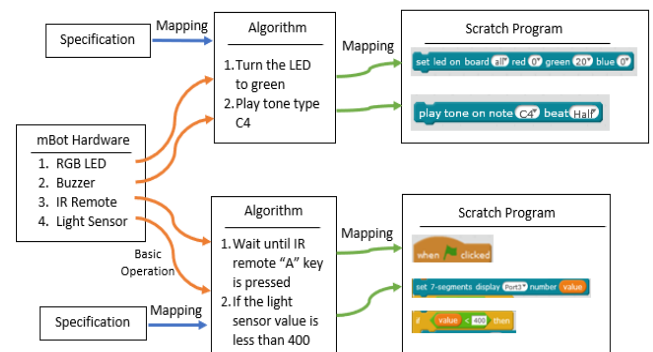


Figure 5 Toolkit Organization

3.2 Basic Principle

The toolkit is useful to learn algorithmic thinking in various ways. A toolkit user can understand the purpose and the behavior of a scratch program by reading the corresponding specification and algorithm even if the user does not know scratch programming language. A user can understand basic operations of each hardware components from the mapping between mBot hardware component and basic algorithm steps.

Then an algorithm and the corresponding code can be a good sample to develop algorithm and program that the user wants. A user can start developing a scratch code by considering algorithm. The algorithm can be easily converted to a scratch code as long as the algorithm follows certain guidelines described in Section 3.3. The activity of understanding, searching and reusing the toolkit elements will naturally develop ability of algorithmic thinking of the user. Since an algorithm is simpler than the corresponding scratch code, a user can make challenge to a more complex program. This implies that algorithmic thinking will facilitate creativity of the user.

3.3 Algorithm Guidelines

Our algorithm guidelines are defined based on the guidelines defined by Steve McConnell [4].

An algorithm is a sequence of steps. Each step is either a basic operation, a routine call (high level operation), a conditional statement or an iterative statement. Each step must clearly describe intention of the step. Each basic operation must correspond to an mBot hardware component. Each routine call must correspond to a subroutine and each subroutine must correspond to a single high-level operation. Each sequence of conditional statements must cover all possible cases. An iterative statement must clearly describe the operations or the processed data at each iteration.

4. Detailed Explanation of Toolkit Element

We developed a rich set of mBot toolkit using mBlock and run on mBot robot as listed below. We shall demonstrate some of them in this section.

- Basic Code to Understand Each Code Block in mBlock
- Rectangle Drawing
- Hexagon Drawing
- Circle Drawing
- Pentagon Drawing
- Obstacle Detection
- Line Tracing
- Dancing Robot with Light Sensor
- Two mBot Robots Run in a Relay Race
- Six-legged Beetle Robot
- Six-legged Mantis Robot
- Six-legged Crazy Frog Robot
- Detecting Motion Around the Robot
- Control Robot’s Speed with Potentiometer
- Basic Code to Get Potentiometer Value
- Basic Code to Get Joystick Value
- Basic Code to Get Light Sensor Value
- Basic Code to Get Seven Segment Value
- Basic Code to Get Motion Sensor Value
- Basic Code to Get RGB LED Value

4.1 Rectangle Drawing

4.1.1 Specification

This program makes an mBot robot drawing a rectangle using the motor device as a control.

4.1.2 Algorithm Description

1. Wait until the green flag is clicked

2. Initialize variables
3. Wait until the on-board button is pressed
4. Repeat the following steps 4 times
 - 4.1 Move forward to the indicated distance (10cm)
 - 4.2 Turn Right at an angle of 90 degrees

The above algorithm contains basic operations of the motor device. The correspondence between the device and basic operation is represented in Table 1, Such representation of mBot hardware devices and basic operations can be converted into the building blocks of a program for mBot.

Table 1 Mapping of Device and Basic Operation for Rectangle Drawing

Device	Basic Operation
Motor	4.1 Move forward to the indicated distance (10cm)
	4.2 Turn Right at an angle of 90 degrees

4.1.3 Scratch Code

Each step of the algorithm described in Section 4.1.2 corresponds to a sequence of scratch code as illustrated in Table 2. The entire program is represented in Figure 6. The blue ‘define’ block is used to define a subroutine. The ‘Initialize’, ‘stop’, ‘turnRightby’ and ‘moveForward’ are the procedures to control the motors on mBot robot by initializing variables for the calculation of the Motor value. The readers can observe that each scratch code corresponds to exactly one algorithm step.

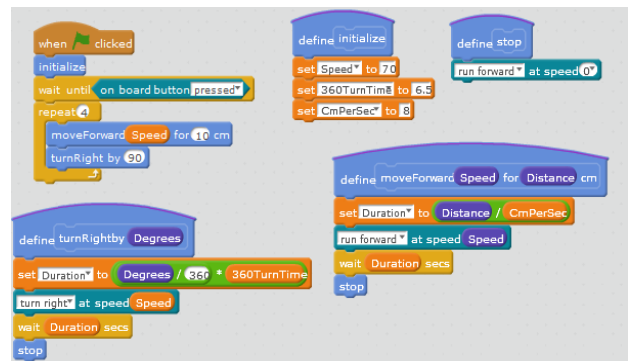

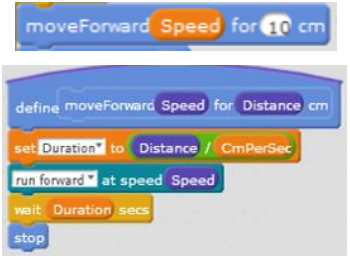
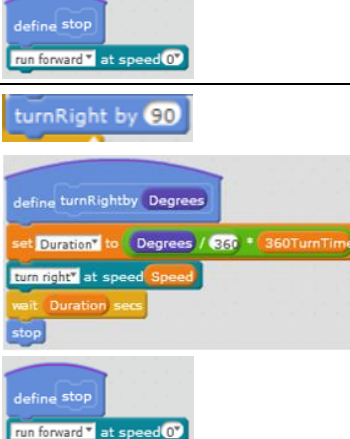


Figure 6 Scratch Code: Rectangle Drawing

Table 2 Mapping between Basic Operation and Scratch Code

Basic Operation	Scratch Code
1. Wait until the green flag is clicked	
2. Initialization variable	
3. Wait until the on-board button is pressed	

Basic Operation	Scratch Code
4. Repeat the following steps 4 times	
4.1 Move forward to the indicated distance (10cm)	
4.2 Turn Right at an angle of 90 degrees	

4.2 Obstacle Detection

4.2.1 Specification

This program is to control the mBot robot to avoid an obstacle. The robot is traveling on a straight line and detects the distance of the obstacle and avoids it to continue the travel. The robot detects the distance of the obstacle by using the ultrasonic sensor. We assume that the size of the obstacle is at most 5 cm in diameter.

4.2.2 Algorithm Description

- 1 Wait until the green flag is clicked
- 2 Initialize constants
- 3 Repeat the following steps forever
 - 3.1 Read ultrasonic sensor value to identify the distance to an obstacle
 - 3.2 Run Forward
 - 3.3 If the detected distance is less than 10cm, then do the following steps
 - 3.3.1 Turn Right at 90 degrees
 - 3.3.2 Move Forward to 10cm
 - 3.3.3 Repeat the following steps 2 times
 - 3.3.3.1 Turn Left at 90 degrees
 - 3.3.3.2 Move Forward to 10cm
 - 3.3.4 Turn Right at 90 degrees

Table 3 Mapping of Device and Basic Operation for Obstacle Detection

Device	Basic Operation
Ultrasonic Sensor	3.1 Read ultrasonic sensor value to identify the distance to an obstacle
	3.3 If the detected distance is less than 10cm, then do the following steps
Motor	3.2 Run Forward
	3.3.2 and 3.3.3.2 Move Forward to 10cm
	3.3.1 and 3.3.4 Turn Right at 90 degrees
	3.3.3.1 Turn Left at 90 degrees

4.2.3 Scratch Code

The entire program for obstacle detection can be found in Figure 7. Each step of the algorithm described in Section 4.2.2 corresponds to a sequence of the program. The readers can easily find correspondence between each algorithm step and the scratch code so that we shall omit the detailed explanation also because of the space limitation,

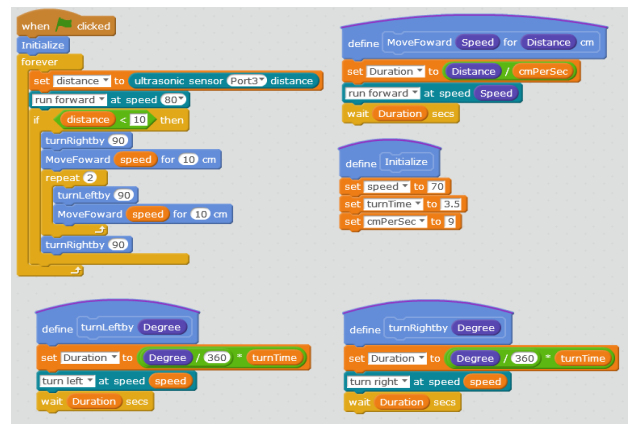


Figure 7 Scratch Code: Obstacle Detection

4.3 Line Tracing

4.3.1 Specification

This program is to control mBot robot to follow the black line. No matter what the shape of the black line, the robot will follow the line. The line follower sensor is located at the lower front of the robot. This sensor has two LED blue light at the left and right sides of the sensor to indicate the sensor value. The sensor value will be used to control mBot robot.

4.3.2 Algorithm Description

- 1 Repeat the following steps forever
 - 1.1 If IR remote A key is pressed, do the following steps
 - 1.1.1 Read line follower value
 - 1.1.2 If the line follower is on the line, then do the following steps
 - 1.1.2.1 Go straight
 - 1.1.2.2 Turn the LED to red
 - 1.1.3 If the line follower is on the right side of the line, then do the following steps
 - 1.1.3.1 Turn left
 - 1.1.3.2 Turn the LED to green
 - 1.1.4 If the line follower is on the left side of the line, then do the following steps

- 1.1.4.1 Turn right
- 1.1.4.2 Turn the LED to blue
- 1.1.5 If the life follower is outside of the line, then do the following steps
 - 1.1.5.1 Go backward
 - 1.1.5.2 Turn the LED to white
- 1.2 Otherwise the Robot will stop

Table 4 Mapping of Device and Basic Operation for Line Tracing

Device	Basic Operation
LED	1.1.2.2 Turn the LED to red
	1.1.3.2 Turn the LED to green
	1.1.4.2 Turn the LED to blue
	1.1.5.2 Turn the LED to white
Line Follower Sensor	1.1.1 Read line follower value
	1.1.2 If the line follower is on the line, then do the following steps
	1.1.3 If the line follower is on the right side of the line, then do the following steps
	1.1.4 If the line follower is on the left side of the line, then do the following steps
	1.1.5 If the life follower is outside of the line, then do the following steps
IR Remote Controller	1.1 If IR remote A key is pressed, do the following steps
Motor	1.1.2.1 Go straight
	1.1.3.1 Turn left
	1.1.4.1 Turn right
	1.1.5.1 Go backward
	1.2 Otherwise the Robot will stop

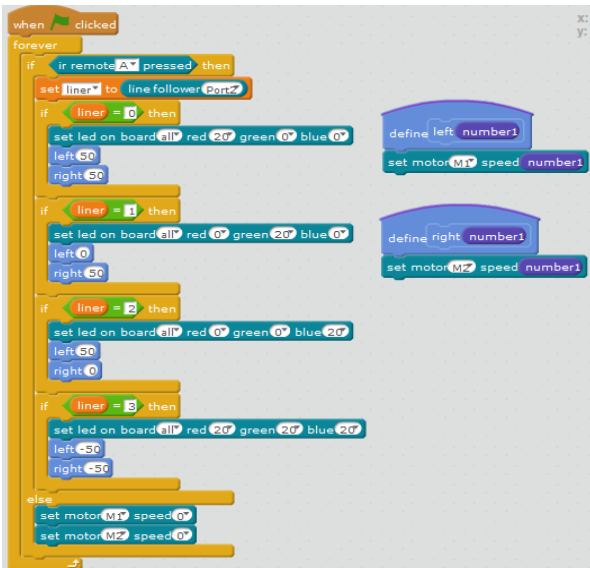


Figure 8 Scratch Code Line Tracer

4.3.3 Scratch Code

The entire program is represented in Figure 8. In this case the blue 'define right(number1)' has function to set Motor M2 speed. 'Right' is the name of variable and 'number1' is a parameter. Then, if variable 'right' is called, it will set the

motor (M2) according to the parameter.

Implementation Notes

- Line follower Sensor is on Port2
- M1 motor: left wheel
- M2 motor: right wheel
- Line trace sensor values are defined as illustrated in Table 5

Table 5 Value of the Line Tracer Sensor

Value	State Description
0	Line follower is on the line
1	Line follower is on the right side of the line
2	Line follower is on the left side of the line
3	Line follower is not on the line

4.4 Six-legged Beetle Robot

4.4.1 Specification

This program is to control Six-Legged Beetle Robot run forward and run backward which controlled by IR Remote controller (Figure 9).

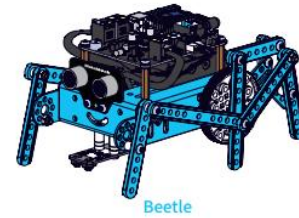


Figure 9 Six-legged Beetle Robot

4.4.2 Algorithm Description

- 1 Repeat the following step forever
 - 1.1 If IR remote "up arrow" key is pressed, do the following steps.
 - 1.1.1 Turn the LED to red
 - 1.1.2 Turn the left wheel with speed 255
 - 1.1.3 Turn the right wheel with speed 255
 - 1.2 Otherwise do the following steps
 - 1.2.1 If IR remote "down arrow" key is pressed, do the following steps.
 - 1.2.1.1 Turn the LED to green
 - 1.2.1.2 Turn the left wheel backward with speed 255
 - 1.2.1.3 Turn the right wheel backward with speed 255
 - 1.2.2 If IR remote "A" key is pressed, do the following steps.
 - 1.2.2.1 Turn the LED to blue
 - 1.2.2.2 Stop turning the left wheel
 - 1.2.2.3 Stop turning the right wheel

Table 6 Mapping of Device and Basic Operation for Six-Legged Beetle Robot

Device	Basic Operation
LED	1.1.1 Turn the LED to red
	1.2.1.1 Turn the LED to green
	1.2.2.1 Turn the LED to blue
Motor	1.1.2 Turn the left wheel with speed 255

Device	Basic Operation
	1.2.1.2 Turn the left wheel backward with speed 255
	1.2.2.2 Stop turning the left wheel
	1.1.3 Turn the right wheel with speed 255
	1.2.1.3 Turn the right wheel backward with speed 255
	1.2.2.3 Stop turning the right wheel
IR Remote Controller	1.1 If IR remote “up arrow” key is pressed, do the following steps.
	1.2.2 If IR remote “A” key is pressed, do the following steps

4.4.3 Scratch Code

Each step of the algorithm described in Section 4.4.2 corresponds to a sequence of scratch code and the entire program is represented in Figure 10.

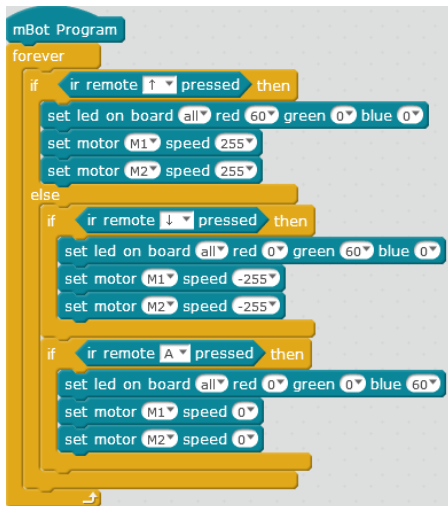


Figure 10 Scratch Code of Six-legged Beetle Robot

4.5 Dancing Robot with Light Sensor

4.5.1 Specification

This program is to make robot dancing when the intensity value of light sensor less than 400 and stop dancing when the intensity of light sensor more than 400. The mBot needs the following two types of electronic modules (Figure 11).

- **Light Sensor** is used to detect the light intensity. Value of this sensor $0 < 1000$
- **Seven Segments** Serial Display is a classic 4-digit, it usually used to display numbers and a few special characters or we can easily use it to show speed, time, the value of sensors, or scores.



Figure 11 mBot with Light Sensor and Seven Segments

4.5.2 Algorithm Description

- 1 Repeat the following step forever
 - 1.1 Turn the LED on board off
 - 1.2 Set robot’s speed 0
 - 1.3 Set variable value with light sensor in port 4
 - 1.4 Set Seven Segments display in port 3 with the value of light sensor
 - 1.5 If the light sensor value is less than 400, do the following steps
 - 1.5.1 Turn the LED on board to red
 - 1.5.2 Run forward with speed 100
 - 1.5.3 Wait for 0.5 secs
 - 1.5.4 Turn the LED on board to green
 - 1.5.5 Run backward with speed 100
 - 1.5.6 Wait for 0.5 seconds
 - 1.5.7 Turn the LED on board to blue
 - 1.5.8 Turn the right wheel with speed 100
 - 1.5.9 Wait for 0.5 seconds
 - 1.5.10 Turn the LED on board to blue and green
 - 1.5.11 Turn the left wheel with speed 100

Table 7 Mapping of Device and Basic Operation for Dancing Robot with Light Sensor

Device	Basic Operation
Light Sensor	1.4 Set Seven Segments display in port 3 with the value of light sensor
	1.5 If the light sensor value is less than 400, do the following steps
Motor	1.2 Set robot’s speed to 0
	1.5.2 Run forward with speed 100
	1.5.5 Run backward with speed 100
	1.5.8 Turn the right wheel with speed 100
LED on Board	1.5.11 Turn the left wheel with speed 100
	1.1 Turn the LED on board off
	1.5.1 Turn the LED on board to red
	1.5.4 Turn the LED on board to green
	1.5.7 Turn the LED on board to blue
Seven Segments	1.5.10 Turn the LED on board to blue and green
	1.4 Set Seven Segments display in port 3 with the value of light sensor

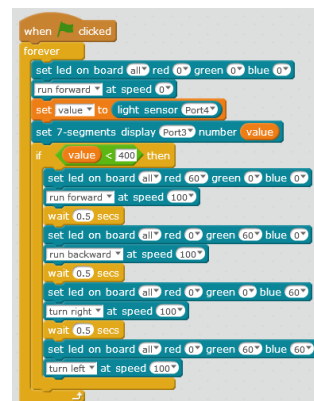


Figure 12 Scratch Code of Dancing Robot with Light sensor

4.5.3 Scratch Code

Each step of the algorithm described in Section 4.5.2

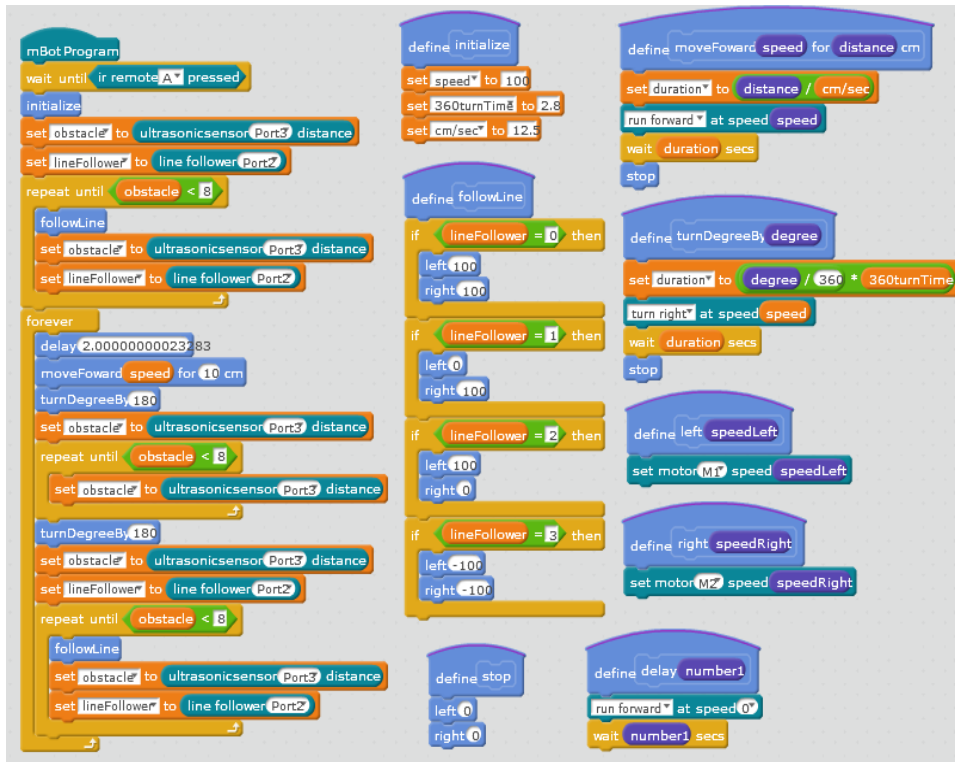


Figure 15 Scratch Code of 1st mBot Robot

corresponds to a sequence of scratch code and the entire program is represented in Figure 12.

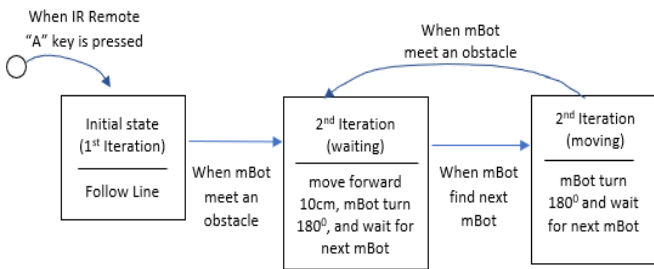


Figure 13 State Transition Diagram of 1st mBot Robot

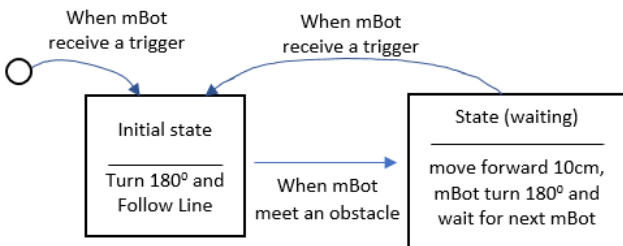


Figure 14 State Transition Diagram of 2nd mBot Robot

4.6 Two mBot Robots Run in a Relay Race

4.6.2 Specification

This program will run in an infinite loop. The first robot will run by following the line and if there's an obstacle 1st mBot will delay for 2sec (to wait for the 2nd mBot run) and 1st mBot run forward to 15cm and turn 180 degrees (to receive trigger from 2nd mBot), 1st mBot will turn 180 degrees again after receive the trigger and following the line. And this algorithm also used in the 2nd mBot but the different is the trigger which make the robot move for first time and 2nd mBot just looping in

2 states in each iteration. For the detailed, see the state transition diagrams (Figures 13 and 14).

4.6.2 Algorithm Description

The state transition diagrams can be represented by the following algorithms.

Algorithm for 1st mBot Robot:

- 1 Wait until IR remote "A" key is pressed
- 2 Initialize variables
- 3 Follow the line until there is an obstacle within 8cm
4. Repeat the following steps forever
 - 4.1 Wait for 2.0 second
 - 4.2 Move forward to 10 cm
 - 4.3 Turn 180 degrees
 - 4.4 Wait until there is an obstacle within 8cm
 - 4.5 Turn 180 degrees
 - 4.6 Follow the line until there is an obstacle within 8cm

Algorithm for 2nd mBot Robot:

1. Initialize variables
2. Repeat the following steps forever
 - 2.1 Wait until there is an obstacle within 10cm
 - 2.2 Turn 180 degrees
 - 2.3 Follow the line until there is an obstacle within 10cm
 - 2.4 Wait for 2.0 second
 - 2.5 Move forward to 10 cm
 - 2.6 Turn 180 degrees

Table 8 Mapping of Device and Basic Operation for Two mBot Robot in a Relay Race

Device	Basic Operation
Motor	4.2 Move forward to 10 cm
	4.3 and 4.5 Turn 180 degrees
Ultrasonic Sensor	4.4 Wait until there is an obstacle within 8cm
Line Tracer Sensor	3 and 4.6 Follow the line until there is an obstacle within 8cm
IR Remote Controller	1. Wait until IR remote "A" key is pressed

4.6.3 Scratch Code

Figure 15 implements the algorithm for the first mBot. The scratch code for the second mBot can be developed similarly.

5. Preliminary Evaluation

We asked for a review of our toolkit to a secondary school teacher teaching computer programming and a university professor who is an advisor of the teacher. They said that it is important to present couple of examples to the students which represent intermediate concept between program and specification as well as basic operations of mBot hardware and sample programs. Then secondary school students will understand and utilize the toolkit to create new application. They evaluated sample algorithms defined in the toolkit is appropriate as such intermediate concept. They also mentioned that the description of sample algorithms should be flexible enough so that students can develop various new algorithms without too much restriction of the description style.

6. Related Work

Although there is some work of computational thinking education, little is known about algorithmic thinking education.

Sullivan, A., et al. [6] proposed a computational thinking education utilizing playful learning with KIBO. The concept is: 1) algorithms, 2) modularity, 3) control structures, 4) representation, 5) hardware/software, 6) the design process, and 7) debugging. Through interlocking wooden programming blocks that contain no embedded electronics or digital components (KIBO), students can explore logical organization and sequencing using the tangible programming blocks which they call algorithms.

Phetsrikran, T., et al. [5] proposed the first step in teaching computational thinking through mobile technology and robotics. This paper provides an application containing puzzles that have different standard difficulty and topic in each level. Through this program can be installed on iPad and connected to a robot, students must send a command and control the robots via Bluetooth to solve the puzzle.

From our viewpoint, Sullivan, A., et al. [6] focus on programming, not an algorithm, since they do not distinguish algorithm and programming. Phetsrikran, T., et al. [5] try to teach pure programming and do not teach computational

thinking because the children are required to solve the puzzle in the form of a computer program. Compared to these works, our toolkit focuses on algorithmic thinking by providing a collection of specifications, algorithms and scratch programs as demonstrated in Section 4.

7. Concluding Remarks

We proposed a toolkit to teach algorithmic thinking for mBot robot programming in this paper. The toolkit is composed of specification, basic mBot operations, algorithm, scratch programs and a set of relationships among them. We demonstrated the effectiveness of the toolkit using various examples. The concept of our toolkit can also be utilized to a robot programming other than mBot. This indicates generality of our approach.

Currently the only problem is that the toolkit is described only in English so that there is a language barrier for most of the Japanese secondary school students. We are planning to develop a Japanese version of the toolkit as a future work for evaluation at an actual class.

Acknowledgment

We appreciate Professor Sumi Kazuhiro at Faculty of Education, Saga University and Ms. Nishiyama at Jonan Junior High School for the preliminary evaluation of our toolkit.

Reference

[1] "Arduino Home Page". <https://www.arduino.cc/en/Guide/Introduction>, (accessed on 2018-01-11).

[2] Curtis, S., "Two thirds of parents 'don't know what an algorithm is'", <http://www.telegraph.co.uk/technology/news/11068502/Two-thirds-of-parents-dont-know-what-an-algorithm-is.html>, 2014-09-02, (accessed on 2018-02-05).

[3] Liao Y., Zhao T., "mBlock Kids maker rocks with the robots", 2013". <http://www.mblock.cc/edu/mblock-kids-maker-rocks-with-the-robots/>, (accessed on 2018-01-11).

[4] McConnell, S., "Code Complete", Second Edition, Microsoft Press, 2004.

[5] Phetsrikran, T., Massagram, W., & Harfield, A., "First steps in teaching computational thinking through mobile technology and robotics", *Asian International Journal of Social Sciences*, 17(3), 37-52, 2017.

[6] Sullvan, A., Bers, M., Mihm, C., "Playing, and Coding with KIBO: Using Robotics to Foster Computational Thinking in Young Children", in Proc. CTE 2017 (International Conf. on Computational Thinking Education), pp. 110-115, 2017.

[7] Teppattra, S., "What possibilities and hindrances do mathematics and technology teachers express regarding teaching programming in compulsory school?", *Ämneslärarexamen*, 225 hp, Kompletterande pedagogisk utbildning, Ange datum för slutseminarium 2017-03-20.

[8] Wang Y., "Getting started with mblock, 2013". <http://www.mblock.cc/docs/the-getting-started-guide/>, (accessed on 2018-01-11).