Adv Mobile Learn Educ Res, 2022, 2(2): 443-454
DOI: 10.25082/AMLER.2022.02.013

RESEARCH ARTICLE

Experimental commands development for LEGO WeDo 2.0 in Python language for STEAM robotics advanced classes

Evangelia Anastasaki[1*] **Kostas Vassilakis**[1]

[1] Department of Electrical and Computer Engineering, Hellenic Mediterranean University, Crete, Greece

Correspondence to: Evangelia Anastasaki, Department of Electrical and Computer Engineering, Hellenic Mediterranean University, Crete, Greece;
Email: eveanast@gmail.com

Received: June 12, 2022;
Accepted: August 3, 2022;
Published: August 9, 2022.

Citation: Anastasaki, E., & Vassilakis, K. (2022). Experimental commands development for LEGO WeDo 2.0 in Python language for STEAM robotics advanced classes. *Advances in Mobile Learning Educational Research*, **2**(2), 443-454.
https://doi.org/10.25082/AMLER.2022.02.013

Abstract: In STEAM education, Lego WeDo 2.0 robot kit is a well-known tool for introducing educational robotics in elementary schools. This kit teaches students the skills necessary for future success. It provides a wide array of educational opportunities across subjects, along with lessons and other digital resources. This article presents experimental commands/functions development in Python programming language through a Raspberry Pi, permitting a suitable connection to the Lego WeDo 2.0 robot based on Scratch WeDo 2.0 commands for STEAM robotics learning in advanced classes. The main reasons for developing the commands are that Scratch language is a novice programming, and students gain incorrect perceptions of programming behaviour. In contrast, Python is real-world programming, in which students can utilise the language in future careers, and students can also create dynamic programs in Python using WeDo 2.0. Additionally, in this study, some projects are presented using the constructed Python functions developed by us versus the same programs in Scratch as examples for activities in the STEAM classrooms using Lego WeDo 2.0 Robot Kit. The limitation of this study was the lack of testing functions in actual instructive practice for data collection about the effectiveness of Python WeDo 2.0 commands in the classroom. The contribution of this study lies in the novelty framework of the development of WeDo 2.0 Python functions, which can be utilised in STEAM robotics advanced classrooms for learning in the fields of science, technology, engineering, the arts and mathematics.

Keywords: STEAM, educational robotics, Scratch, Python, WeDo 2.0, Raspberry Pi

1 Introduction

Traditional programming methods do not solve students' problems, resulting in negative attitudes towards programming (Owen-Hill, 2021). STEAM education solves the problems created by conventional teaching approaches as it is an effective tool, and it contributes forward cultivating and developing cognitive structures by students (Yakman, 2008). STEAM is an approach to learning that employs Science, Technology, Engineering, the Arts, and Mathematics to direct pupil inquiry, dialogue, and critical thinking (Korkmaz, 2016). "Scratch MIT" is a visual programming language that allows children to effectively create interactive content (Korkmaz, 2016; Papadakis ET AL., 2016). "Python" is a "real" programming language with valuable data structures as a core part of the language; it gives them prominent names and makes them extraordinarily easy to use (Vega, 2018; Verawati et al., 2022). Such usefulness, simplicity, and clarity remove barriers to programming, making the transition from Scratch language smoother (Ladias et al., 2021; 2022). The development set WeDo 2.0 permits pupils to construct and program basic LEGO models connected to a computer and can execute essential robotic assignments. This practice contains two stages, software, and hardware, which encourage the introduction of educational robotics into the classroom (Chalmers, 2018; Olabe et al., 2011; Papadakis, 2020). Programming can be too difficult and tedious when learned through the conventional "abstract" method (Kapaniaris & Zampetoglou, 2021). On the contrary, pupils learn what robots can and cannot do with direct experience and understanding by handling a physical robot and noticing its situation (Lazarinis et al., 2022). By programming robots, pupils can discover if their aptitudes and interests correspond to those skills that will define the future job market, like programming, science, technology, or engineering (Kalovrektis et al., 2021; Tzagkaraki et al., 2021).

This research presents the transition from Scratch to Python programming using the LEGO WeDo 2.0 robot kit by implementing python commands/functions. Notably, some commands were designed and implemented based on Scratch MIT and Python using the LEGO WeDo 2.0 robot kit for advanced pupils to shift from Scratch to Python programming. The main objective of the research was to have Lego WeDo 2.0 operates with Python compared to Scratch for robotics learning.

2 Theoretical framework

2.1 STEAM & educational robotics

STEAM is an integrated learning approach that requires a link between standards, assessments, and course design/implementation. STEAM experiences include standards from Science, Technology, Engineering, Mathematics, and the Arts that must be taught and evaluated together (Bertrand & Namukasa, 2020). STEAM is an evolution of the original acronym STEM, plus one extra element: art. The art could offer the students "socioemotional" and "holistic development" (Ampartzaki et al., 2022). Integrating the skills into STEM learning has enabled teachers to extend the benefits of hands-on training and collaboration in various areas, fostering creativity and curiosity at the core (Kalogiannakis & Papadakis, 2017a, 2017b; Papadakis, 2020).

Seymour Papert, in 1969, developed the Logo programming language and "robot turtles". Papert considers that people learn according to mental models and learning environments (Catlin & Woollard, 2014). Papert's "turtles" are constructions programmed by learners exploring the world around them. The turtle is an "educational robot" on the ground that moves based on commands given by the computer. Based on Logo, various environments have been developed, called "logo-like environments", and they are also used in physical computing, such as Scratch, EV3/Lego programming, etc. (Catlin & Woollard, 2014; Kalovrektis et al., 2021).

Educational robotics within STEAM frameworks can attain an attractive strategy that changes boring concepts into a fun learning preparation. STEAM robotics instruction gives imaginative challenges and openings for school-level learners to create one-of-a-kind concepts and advanced learning skills (Afari & Khine, 2017). STEAM with educational robotic kits is an efficient approach because they encourage the ease with which pupils can connect among STEAM disciplines (Plaza et al., 2020; Plaza et al., 2018). Robotic kits like LEGO's, connected with a block-based programming language like Scratch, can help the pupil's fundamental programming learning (Dorling & White, 2015; Ruzzenente et al., 2012). The LEGO robot kits are appropriate tools that include pupils with hands-on learning through building with LEGO bricks and programming in a block-based programming environment (Chalmers, 2018; Elkin et al., 2014), such as Scratch or with text-based programming (Armoni et al., 2015; Weintrop & Wilensky, 2017), such as Python. Educational Robotics with Python Language is used for teaching at various levels of education (Blank et al., 2003; Khamphroo et al., 2017; Vega, 2018). Many research articles reported positive results of using robots with Python to aid students in understanding programming concepts and writing programs (Khamphroo et al., 2017; Weintrop & Wilensky, 2017).

2.2 Learning theories & skills

Learning theories applicable in the development of LEGO WeDo 2.0 are constructionism, active learning, computational thinking, creative thinking, and problem-solving.

2.2.1 Constructionism

WeDo 2.0 development is rooted in the theory of constructionism, which says pupils learn to verbalise their opinions clearly and collaborate on tasks successfully by sharing in group projects (Evripidou et al., 2020). Papert's constructionism focuses on learning in the circumstances "rather than looking at them from a distance, that connectedness rather than separation are powerful means of gaining understanding" (Ackermann, 2001). This implies that new experiences are the entirety of a single experience made by applying existing knowledge to enhance it. In this manner, "hands-on exercises are the best for the classroom applications of constructionism, critical considering and learning" (Ackermann, 2001). Each person has a different perception and construction of the knowledge process (Khanlari, 2014).

2.2.2 Active learning

WeDo 2.0 development approximates active learning putting a more prominent degree of obligation on the student than passive approaches such as lectures. However, educators' direction is still significant within the active learning classroom (Zaher & Hussain, 2020). Active learning exercises with WeDo 2.0 may range in length from some minutes to entire course sessions or take place over numerous course sessions (Zaher & Hussain, 2020).

2.2.3 Computational thinking

WeDo 2.0 development boosts computational thinking through coding activities, which bring students' creations to life, generating motivation and the desire to discover more. It allows the students to analyse a complex problem, understand it and develop possible solutions. Computational thinking is a set of aptitudes to solve problems (Wing, 2008), it is a thought process (Psycharis, 2018), or it is a problem-solving process (Papadakis & Kalogiannakis, 2022). So we manage to

present these solutions so that the computer and its operator can understand the problem to look for ways to solve it. The four basic steps in the process of applying computational thinking are (Psycharis, 2018):

(1) Decomposition: breaking down a complex and challenging problem into smaller ones that can be more manageable.

(2) Identification of similarities: the search for similarities between minor problems.

(3) Abstract process: focus only on important information, and ignore irrelevant details.

(4) Algorithms: develop a solution to the problem (step by step) or classify the rules.

2.2.4 Creative thinking

WeDo 2.0 development increases creative thinking, which can interpret as thinking that can connect or see things from a new perspective. Creative thinking boosts curiosity, being resourceful, having the desire to find, enjoying solving problems, having the dedication to work, thinking flexibly, asking lots of questions, giving better answers, being able to synthesise, see new implications, and have a broad knowledge (Murcia et al., 2020). Therefore, generating new ideas and ways to produce a project with WeDo 2.0 leads to gaining new insights, approaches, perspectives, and practices when dealing with issues (McGregor, 2007). It is a skill in carrying out a mindset based on a deep understanding of the concepts that an individual has previously mastered.

2.2.5 Problem-solving

WeDo 2.0 development promotes a problem-solving approach, which is fundamental in numerous sciences, math, and engineering classes. The skill of problem-solving can reflect how somebody resolves issues precisely and reasonably (Astuti et al., 2021). Effective problem-solving approach premises four basic steps (Huei, 2015):

(1) Define the problem: Diagnose the situation so that your focus is on the problem.

(2) Generate alternative solutions: Brainstorm on others' ideas.

(3) Evaluate and select an alternative: Estimate alternatives relative to established goals.

(4) Implement the solution: Plan and implement a pilot test of the chosen alternative.

3 Methodology

3.1 Development research design

To operate LEGO WeDo 2.0 with Python for smoothly moving from Scratch to Python programming language for teaching advanced STEAM robotics. WeDo 2.0 is an applied science toolkit for students that develops programs on a computer or smart device based on a connected robot kit utilising Bluetooth Low Energy (BLE) technology (Chalmers, 2018; Olabe et al., 2011). WeDo 2.0 Robot kit includes 280 LEGO blocks and electronics such as Smarthub, Motor, Tilt Sensor, and Distance Sensor. A study strategy was utilised as a focused research design to develop the Python WeDo 2.0 commands based on Scratch WeDo 2.0 commands; we initially studied Scratch Language's WeDo 2.0 block commands. Then, we compared the Scratch and Python WeDo 2.0 commands presenting some applied examples with the robot kit of WeDo 2.0.

3.2 Research procedure

Interacting with Lego WeDo 2.0 with Python required a device with Bluetooth Low Energy (BLE) technology, such as Raspberry Pi. We utilised Raspberry Pi and Python GATT Library "Gattlib" to interact with WeDo's BLE system in this research.

The Raspberry Pi is a low-cost, credit-card-sized computer (Raspberry, n.d.). "Gattlib" is a python library (Linux only) to access Bluetooth LE devices. It utilises the command "gatttool" within the "BlueZ" package. BlueZ is a Linux operating system software that actualises the Bluetooth protocol stack. So, the gatttool in "Gattlib" can only be utilised on Linux computers.

GATT (Generic Attribute Profile) is a component of the host. This system determines the BLE-based information model and the related methods by which devices can find, read, and adjust each other's information. GATT is built based on "addressable" information (ATT protocol) within the shape of attributes (Leonardi et al., 2018). The attribute is an information entity that holds data around itself and any value available through a one-of-a-kind "handle", depending on the get-to rights of the given point. GATT makes a hierarchy that composes the subtle elements and creates a solid system for building GATT-based profiles (Leonardi et al., 2018).

Also, the connection to the Raspberry Pi Platform is realised through "Headless Mode" with a direct ethernet connection. Headless Mode is a mode that can directly access the terminal or the Desktop of Raspberry Pi through VNC over the network (Wi-Fi or Ethernet) without needing a monitor, keyboard or mouse (Dexter, n.d.)

Based on the previously described testbed, we constructed experimental commands in Python's programming language, permitting a suitable connection to the WeDo 2.0 robot using functions (Motor Activation, RGB LED Color Change, Sound Playing, Distance Sensor Activation, Tilt Sensor Activation). This would allow STEAM educational robotics to be taught with Python utilising WeDo 2.0.

Therefore, in developing Smarthub (or "brain" of the robot) commands, it was necessary to know more precisely how the commands are formed in the form of byte sequences. LEGO's GitHub profile provided much information to interact with LEGO Hub through the "LEGO Wireless Protocol (LWP) GATT services".

The LEGO Bluetooth Hub Profile contains a single BLE Generic Attribute Profile (GATT) service. The service lets users get data regarding the LEGO Hub, such as name, battery level, etc., and interact with connected sensors and motors (LegoGitHub, n.d.). Also, a blog of Portuguese origin called "O Falcão" was a great help, where the blog's author had made various attempts through the Command Prompt (CMD) to give commands to WeDo 2.0 and to read values from the sensors (Falcão, 2016).

However, it was vital to discover a Python library that could organise Bluetooth LE communication to begin to construct functions in Python Language. This library is called "Gattlib", which requires a framework with Bluetooth Low Energy (BLE) technology. So, we utilised the Raspberry Pi Platform with the created Python commands and read characteristic values from the sensors to communicate with WeDo 2.0.

3.2.1 Techniques

Before we constructed the functions, we developed particular techniques:

(1) We used the Python module "`pip`" to install the Python GATT library (gattlib). So, we used to type "`pip install gattlib`" at the command prompt of Raspberry pi.

(2) For the Smarthub connection, we created a "`GATTRequester`" object (in coding, an "object" is any entity that has a state (properties or attributes) and behaviour (methods or functions). In our case, the "object" is the Smarthub of WeDo) to read data passing the device address and interface "`hci0`" of WeDo 2.0 to connect. Then, we read a value determined either by its "`handle`" or its "`UUID`". This process assumes that the WeDo 2.0 smart hub is kept close to the computer for maximum signal strength during the connection. Then we pressed the button on Smarthub to connect.

3.2.2 WeDo 2.0 Python functions

For developing WeDo 2.0 Python Functions, we used the method (function) "`write_by_handle`" to send the data.

3.2.2.1 Motor Activation Function

We used to send the data with a sequence of 4 bytes with handle features. The first byte represents the port. This means the value is 01 for Port A or 02 for Port B to construct motor activation functions (turn on/off, turn right/left):

(1) Turn on: In our main program, we need to call the function "`motor_on()`" to turn on the motor.

(2) Turn off: In our main program, we need to call the function "`motor_off()`" to turn off the motor.

(3) Turn right: In our main program, we need to call the function "`motor_that_way()`" to turn the motor right.

(4) Turn left: In our main program, we need to call the function "`motor_this_way()`" to turn the motor left.

3.2.2.2 LED RGB color change function

We used the handle features and 11 bytes for each colour to construct the LED RGB Colour Change Function. The first byte, "06", is the LED port, the second byte is the command sent to the port, so "04" indicates "change colour", and the third byte is the length of the arguments of the command, so "01" means that the colour is just 1-byte length. For the colours of LED, each is a different colour, and they exist 11 values for each colour:

(1) 00 → Off
(2) 01 → Pink
(3) 02 → Purple
(4)03 → Blue
(5) 04 → Cyan
(6) 05 → Light Green
(7) 06 → Green

(8) 07 → Yellow
(9) 08 → Orange
(10) 09 → Red
(11) 0A → Light Blue

For instance, the handle "`06040106`" indicates that the Smarthub's LED changes to Green.

So, in our main program, we need to call the function "`colours()`" to change the colours of the LED.

3.2.2.3 Sound playing function

The sound is controlled by the same handle operated to control the motor and the RGB LED (`0x003d`) to construct the sound playing function. The "port" is "05", and the "command" to trigger the sound is "02", pursued by a payload of "04" bytes covering:

(1) the Frequency in Hz (2 bytes);
(2) the Duration in Ms (2 bytes).

Calling the sound function produces a piezo tone. The piezo tone alerts us when a certain threshold is met.

Therefore, in our main program, we need to call the function "`sound()`" to play a piezo tone.

3.2.2.4 Motion (distance) sensor activation function

The distance sensor is controlled by the same handle operated to control the motor. The RGB LED and the Piezo Tone constructed the motion (distance) sensor activation function by its operating mode value. Python Function based on this formula:

$$\frac{\text{Distance} = \text{Time} \times \text{Speed}}{2}$$

Furthermore, the python library "Pyserial" are utilised to connect to WeDo 2.0 port with raspberry pi. From this library, three functions are used:

(1) open ( ) → open the serial port;
(2) close ( ) → close the serial port;
(3) readline ( ) → read a string from the serial port.

Then, in our main program, we need to call the function "`distance()`" to measure the distance between an object.

3.2.2.5 Tilt Sensor Activation Function

The tilt sensor is controlled by the handle value and its operating mode value to construct the tilt sensor activation function.

Now, in our main program, we need to call the function "`tilt()`" to measure the tilt of an object.

4 Results

This section presents some projects using the constructed Python functions we developed versus the same programs in Scratch to show the use of functions in practice. Based on lesson plan *"Speed"* on Lego Education Website (LegoEducation, n.d.), we created the programs in Scratch and Python below as examples for activities in the STEAM advanced classrooms using Lego WeDo 2.0 Robot Kit (Figure 1).

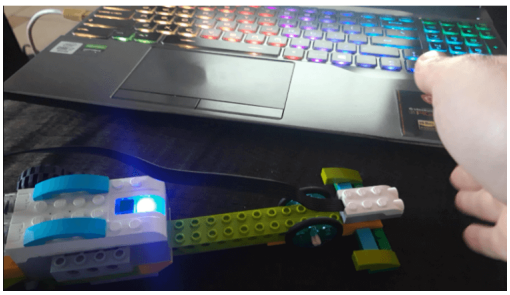

Figure 1 Using LEGO Blocks for car construction in the "Speed" lesson plan framework

For all projects below, we used an infinite loop (or endless loop), a loop that is a sequence of instructions continually repeated until a specific condition is reached. Scratch has a block type for making infinite loops, called a "forever block". There is no equivalent, no clear forever block in Python, but a conditional loop is used, constantly evaluating "True". So, while "True", running this code is equivalent to forever. In addition, we used the "sleep" function in Python code to get

the program's delay. The "sleep" function is imported from the "time" module. In Scratch code, we utilised the block "wait". This block is used to wait for an "n" number of seconds, where "n" is any integer or float. This is known as concurrency.

4.1 Project A: using motor activation functions

In this project, we used the motor activation functions in Python (Figure 2(A)) and Scratch (Figure 2(B)).

To turn on/off the Smarthub, in Python Figure 2(A)), we utilised the "motor_on / motor_off" functions, whereas, in Scratch (Figure 2(B)), we used the "turn on/off" block command, in which we had to set the port with a choice: "Motor A", "Motor B", or "All Motors".

Also, to turn left/right the motor, we used the "motor_this_way / motor_that_way" functions with the parameter "movehub" (which contains the connection with Smarthub of WeDo 2.0) in Python. In contrast, in Scratch, we utilised the "set [Motor A/B/All] direction to that / this way" block command. Clearly, in Scratch, we should set the motor's port and then the motor's direction left or right, contrasting with Python code.

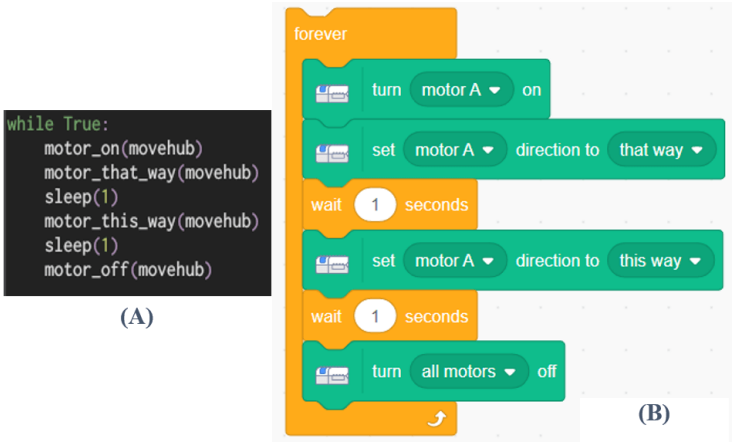

Figure 2 Motor Activation Functions: (A) Python (B) Scratch

4.2 Project B: distance sensor activation function

In this project, we used the distance sensor activation function in Python (Figure 3(A)) and Scratch (Figure 3(B)).

To measure the distance between an object, in Python code (Figure 3(A)), we used the "distance" function with parameters the connection of Smarthub (movehub), the mode of distance sensor (handle) and the port (A or B). Similarly, we utilised the "distance" block command in Scratch code (Figure 3(B)).

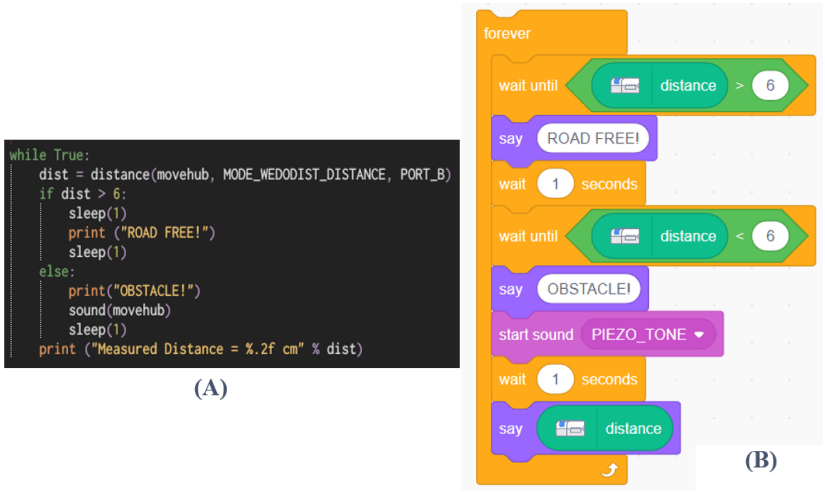

Figure 3 Distance Sensor Activation Function : (A) Python (B) Scratch

Also, in Python code, we used a condition with an "if" statement to control the distance measurement so that if the "distance of the object is bigger than 6" from the robot, output to the screen the message "ROAD FREE!", that is, the road has not some obstacle, differently if

the distance of the object is smaller than 6" from the robot, output to the screen the message "OBSTACLE!", that is, it detected some obstacle. In contrast, in Scratch, we utilised the "wait until" block command, which is a control block. This block pauses its script until the specific "Boolean" condition is "true". In Scratch, the control of distance measurement becomes with similar way to Python code. However, in Scratch, to output to the screen should create a sprite talk by utilising the "say" block command.

Moreover, to produce a piezo tone alerting when an obstacle is detected, in Python, we used the function "sound". In Scratch, we used the "start sound" block command, a Scratch command but not a specific WeDo 2.0 command, such as in Python code.

In addition, at the end of the code in both languages, output to the screen the distance measurement, showing up an "n" number. In Python, the measurement is float numbers (Figure 4(A)), whereas, in Scratch, the numbers are integers (Figure 4(B)).

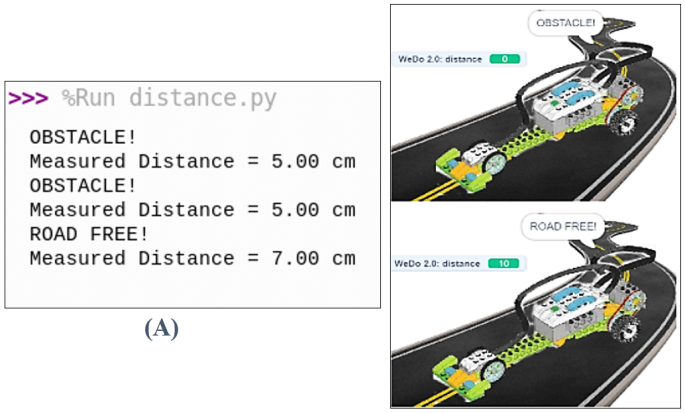

Figure 4 Output to the screen: (A) Python (B) Scratch

4.3 Project C: using tilt sensor activation function

In this project, we used the tilt sensor activation function in Python (Figure 5(A)) and Scratch (Figure 5(B)).

To measure the tilt of the robot, in Python code (Figure 5(A)), we used the "tilt" function with parameters the connection of Smarthub (movehub), the mode of tilt sensor (handle) and the port (A or B). In Scratch code (Figure 5(B)), we utilised the "wait until" block command to activate the "tilt" block command only when the sensor detects some slope.

In addition, when the sensor detects any tilt in both codes, it outputs the message "Tilt!" to the screen.

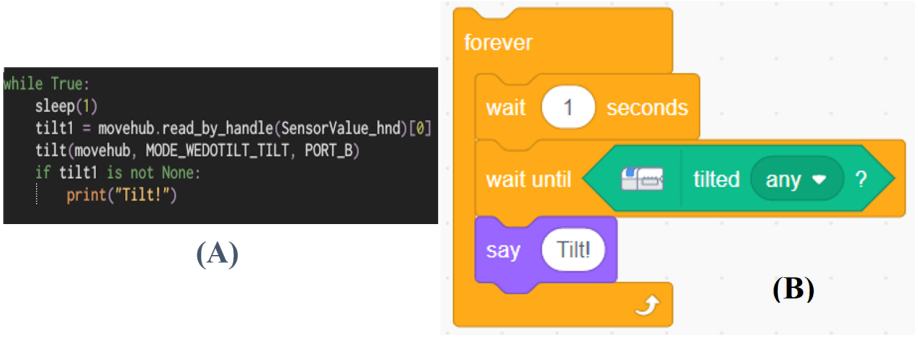

Figure 5 Tilt Sensor Activation Function : (A) Python (B) Scratch

4.4 Project D: LED RGB colour change function

In this project, we used the LED RGB colour change function in Python (Figure 6(A)) and Scratch (Figure 6(B)).

To change the LED colours, in Python code (Figure 6(A)), we used the "colors" function to change the LED colours with the parameter of the connection of Smarthub (movehub). In Scratch code, we utilised the "set light color to" block command.

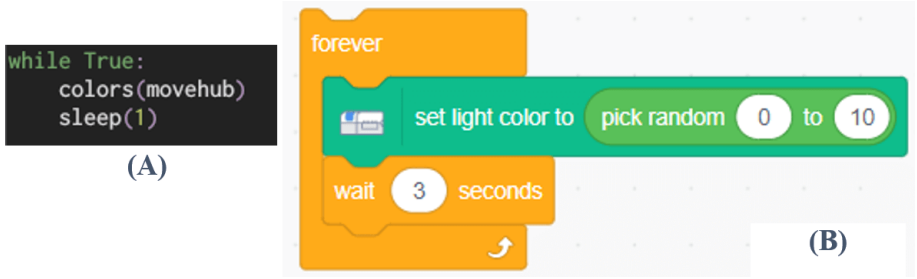

Figure 6 LED RGB Colour Change Function: (A) Python (B) Scratch

Additionally, in Scratch code (Figure 6(B)), we used a further block command, the"pick random 0 to 10", which changes randomly to 11 colours of the LED, contrasting Python code, in which the colours change automated (Figure 7).

Figure 7 Output of the 11 LED Colors of WeDo 2.0 with Python Code

5 Discussion

Through investigations and various experimentations, the lessons with robotic assignments with Lego WeDo 2.0 motivated students, and they also effectively comprehended the fundamental principles of STEAM (Üşengül & Bahçeci, 2020). They are typically done using motorised LEGO models and straightforward programming. WeDo 2.0 bolsters a hands-on, "minds-on" knowledge solution that offers students the confidence to make questions and instruments to discover the answers and solve real-life issues. Students learn by asking questions and solving problems. This material does not tell students everything they need to know. Instead, it makes them question what they know and investigate what they do not yet understand (LegoEducation, 2018). In general, block-based coding, like Scratch, can be a way to introduce coding to students. However, this way is much harder for students to get a handle on actual coding concepts and syntax when dragging and dropping boxes (Moors et al., 2018). With text-based codings, like Python, students get a complete learning involvement that instils essential coding concepts they will keep in mind and can construct. A vital learning perspective is making mistakes, distinguishing what went wrong, and correcting those mistakes (Weintrop & Wilensky, 2019). With Scratch coding, doing and learning from mistakes is not as predominant, as students work inside the boundaries of predefined squares of code and cannot reach those boundaries. In addition, by moving blocks, students learn the common thought of how coding works, though not fundamentally the abilities past that (Armoni et al., 2015).

Meanwhile, Python coding presents numerous real-world challenges that assist students in learning how to solve issues. Since students can adjust and extend past drag-and-drop activities, coding in Python permits students to develop by making mistakes (Weintrop & Wilensky, 2019). Through this learning experience, students will construct the knowledge they ought to code and the adaptability and resilience required for anything they do in school or life.

By understanding the qualities and limits of block-based environments such as Scratch, educators can work towards actualising programming tools most advantageous for beginners and shifting to text-based languages (Weintrop & Holbert, 2017) such as Python. Encouraging and enticing learners in STEAM education, instructors can utilise hands-on projects that move the computational concepts from the screen into the real world to permit learners to interact with them (Juškevičienė et al., 2020). Python can appear to be a complex process, especially for

inexperienced educators. However, the compensation of seeing students learn real-world coding abilities and construct an establishment of information they can apply to any career in the future is well worth the starting challenges.

6 Limitations

The research had limitations, such as testing functions in actual educational practice for extracting conclusions about the effectiveness of constructed commands on STEAM robotic learning. Future research ought to become in the classroom with educators utilizing innovative new tools such as the developed Python functions with WeDo 2.0 to ensure that they are not only developmentally suitable to operate but easy for each educator to instruct. In addition, experts could evaluate the implementation of the commands/functions to STEAM Education, Programming and Pedagogy.

7 Conclusion and future work

In conclusion, as described in the previous sections, the purpose of this work was to support programming beyond the Scratch language.Most schools introduce the software Scratch to cover some fundamental concepts in coding (Papadakis et al., 2020). Scratch certainly is not a coding language in the way most programmers would understand. There are too many essential programming skills they cannot learn on Scratch, but it has some uses for beginners (Papadakis, 2022). However, many schools keep using Scratch to involve the coding framework in the curriculum. Therefore, this harms students for several reasons. Initially, students using novice programming, such as Scratch, gain an incorrect impression of coding because this language is more for beginners and not advanced students (Moors et al., 2018). Over time, students desire to create more complex projects, so Scratch does not cover them (Papadakis & Orfanakis, 2018). Moreover, by learning real-world programming, such as Python, they can apply it in their future careers. For these reasons, experimental commands/functions for Lego WeDo 2.0 in Python were developed for STEAM robotics learning in advanced classes.

Assuming that the students effectively know the comparing and contrasting differences in the language rules of Scratch and Python, it is presented the construction of experimental functions in Python through Raspberry Pi based on Scratch for operating WeDo 2.0. This will assist the students in noticing the abstract concepts of Scratch and Python programming in practice and allow STEAM educational robotics to be taught with Python utilising WeDo 2.0. This development gives (a) a "feedback-oriented learning environment", (b) a "collaborative working environment", and (c) openings to work and investigate arrangements for real-world issues. Furthermore, with educational robots, like WeDo 2.0, students illustrate improvement, show "positive attitudes" toward STEAM, select engineering as a principal, and join an "iterative design method".

Future work should focus on testing the constructed Python functions in STEAM classrooms and extracting conclusions about the effectiveness of the learning procedure and the problem-solving methods students utilised on the different assignments or the kinds of blunders that students made on the programming assignments. This could permit educators to construct suitable activities for advanced students (Papadakis & Kalogiannakis, 2022). Also, it could become an evaluation by a professional with specialised knowledge in the article field (Papadakis et al., 2021).

Moreover, many parts of this work can be upgraded and reconfigured. It could upgrade the tilt function to offer more than one orientation and calculate the angle of these. Another upgrade that can be done is to construct more functions for the motor to control the low/high intensity and to reverse direction.

Conflicts of interest

The authors declare that they have no conflict of interest.

References

Ackermann, E. (2001). Piaget's Constructivism, Papert's Constructionism: What's the difference?

Adafruit, L. S. (n.d.). GATT, Introduction to Bluetooth Low Energy, Adafruit Learning System, 17 March 2022.
https://learn.adafruit.com

Afari, E., & Khine, M. S. (2017). Robotics as an Educational Tool: Impact of Lego Mindstorms. International Journal of Information and Education Technology, 7(6), 437-442.
https://doi.org/10.18178/IJIET.2017.7.6.908

Ampartzaki, M., Kalogiannakis, M., Papadakis, S., & Giannakou, V. (2022). Perceptions About STEM and the Arts: Teachers', Parents' Professionals' and Artists' Understandings About the Role of Arts in STEM Education. Lecture Notes in Educational Technology, 601-624.
https://doi.org/10.1007/978-981-19-0568-1_25

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to 'Real' programming. ACM Transactions on Computing Education, 14(4), 1-15.
https://doi.org/10.1145/2677087

Astuti, N., Rusilowati, A., & Subali, B. (2021). STEM-Based Learning Analysis to Improve Students' Problem Solving Abilities in Science Subject: a Literature Review. Journal of Innovative Science Education, 10(1), 79-86.
https://doi.org/10.15294/jise.v9i2.38505

Bertrand, M. G., & Namukasa, I. K. (2020). STEAM education: student learning and transferable skills. Journal of Research in Innovative Teaching & Learning, 13(1), 43-56.
https://doi.org/10.1108/JRIT-01-2020-0003

Blank, D., Meeden, L., & Kumar, D. (2003). Python robotics: an environment for exploring robotics beyond LEGOs. SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science educationFebruary, 317-321.
https://doi.org/10.1145/611993.611996

Catlin, D., & Woollard, J. (2014). Educational Robots and Computational Thinking. In Proceedings of 4th International workshop teaching robotics, teaching with robotics & 5th International conference robotics in education, 144-151.

Chalmers, C. (2018). Robotics and computational thinking in primary school. International Journal of Child-Computer Interaction, 17, 93-100.
https://doi.org/10.1016/j.ijcci.2018.06.005

Dexter, I. (n.d.). Connecting to Raspberry Pi without a monitor for Beginners, 23 June 2022.
https://www.dexterindustries.com

Dorling, M., & White, D. (2015). Scratch: A way to Logo and Python. SIGCSE 2015 - Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 191-196.
https://doi.org/10.1145/2676723.2677256

Elkin, M., Sullivan, A., & Umashi Bers, M. (2014). Implementing a Robotics Curriculum in an Early Childhood Montessori Classroom. Journal of Information Technology Education: Innovations in Practice, 13, 153-169.
https://doi.org/10.28945/2094

Evripidou, S., Georgiou, K., Doitsidis, L., Amanatiadis, A. A., Zinonos, Z., & Chatzichristofis, S. A. (2020). Educational Robotics: Platforms, Competitions and Expected Learning Outcomes. IEEE Access, 8.
https://doi.org/10.1109/ACCESS.2020.3042555

Falcão, O. (2016). WeDo 2.0 – reverse engineering – O Falcão. Retrieved 15 May 2022.
https://ofalcao.pt

Huei, Y. C. (2015). Benefits and introduction to python programming for freshmore students using inexpensive robots. Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering: Learning for the Future Now, TALE 2014, (December), 12-17.
https://doi.org/10.1109/TALE.2014.7062611

Juškevičienė, A., Stupurienė, G., & Jevsikova, T. (2020). Computational thinking development through physical computing activities in STEAM education. Computer Applications in Engineering Education, 29(1), 175-190.
https://doi.org/10.1002/cae.22365

Kapaniaris, A., & Zampetoglou, G. (2021). Visual programming for the creation of digital shadow play performance using mobile devices in times of Covid-19. Advances in Mobile Learning Educational Research, 1(2), 162-170.
https://doi.org/10.25082/AMLER.2021.02.010

Kalogiannakis, M., & Papadakis, S. (2017a). Pre-service kindergarten teachers acceptance of "ScratchJr" as a tool for learning and teaching computational thinking and Science education. In Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), Research, practice and collaboration in science education (pp. 21-25). Dublin: Dublin City University and the University of Limerick.

Kalogiannakis, M., & Papadakis, S. (2017b). An evaluation of Greek educational Android apps for preschoolers. In proceedings of the 12th Conference of the European Science Education Research Association (ESERA), Research, Practice and Collaboration in Science Education, Dublin City University and the University of Limerick, Dublin, Ireland (pp. 21-25).

Kalovrektis, K., Papageorgiou, T., Psycharis, S., Theodorou Telecommunications, P., Vasiliki Ntourou, G., & Xenakis, A. (2021). The Impact of Physical Computing and Computational Pedagogy on Girl's Self-Efficacy and Computational Thinking Practice. 2021 IEEE Global Engineering Education Conference (EDUCON).
https://doi.org/10.1109/EDUCON46332.2021.9454003

Khamphroo, M., Kwankeo, N., Kaemarungsi, K., & Fukawa, K. (2017). MicroPython-based educational mobile robot for computer coding learning. 2017 8th International Conference on Information and Communication Technology for Embedded Systems, IC-ICTES 2017. https://doi.org/10.1109/ICTEmSys.2017.7958781

Khanlari, A. (2014). Effects of educational robots on learning STEM and students' attitude toward STEM. 2013 IEEE 5th International Conference on Engineering Education: Aligning Engineering Education with Industrial Needs for Nation Development, ICEED 2013, 62–66. https://doi.org/10.1109/ICEED.2013.6908304

Korkmaz, Ö. (2016). The Effect of Scratch- and Lego Mindstorms Ev3-Based Programming Activities on Academic Achievement, Problem-Solving Skills and Logical-Mathematical Thinking Skills of Students. Malaysian Online Journal of Educational Sciences, 4(3), 73-88.

Ladias, A., Karvounidis, T., & Ladias, D. (2021). Classification of the programming styles in scratch using the SOLO taxonomy. Advances in Mobile Learning Educational Research, 1(2), 114-123. https://doi.org/10.25082/AMLER.2021.02.006

Ladias, A., Karvounidis, T., & Ladias, D. (2022). Forms of communications in scratch and the SOLO taxonomy. Advances in Mobile Learning Educational Research, 2(1), 234-245. https://doi.org/10.25082/AMLER.2022.01.007

Lazarinis, F., Karatrantou, A., Panagiotakopoulos, C., Daloukas, V., & Panagiotakopoulos, T. (2022). Strengthening the coding skills of teachers in a low dropout Python MOOC. Advances in Mobile Learning Educational Research, 2(1), 187-200. https://doi.org/10.25082/AMLER.2022.01.003

LegoEducation. (n.d.). (2022). Speed — WeDo 2.0 Lesson Plan — LEGO® Education. https://education.lego.com

LegoEducation. (2018). LEGO® Curriculum Pack. Education WeDo 2.0, 107(2), 45. https://doi.org/10.2169/naika.107.contents2

LegoGitHub. (n.d.). (2022). LEGO Wireless Protocol 3.0.00 Doc v3.0.00 r17 documentation. https://lego.github.io

Leonardi, L., Patti, G., & Lo Bello, L. (2018). Multi-Hop Real-Time Communications over Bluetooth Low Energy Industrial Wireless Mesh Networks. IEEE Access, 6, 26505-26519. https://doi.org/10.1109/ACCESS.2018.2834479

McGregor, D. (2007). Developing Thinking, Developing Learning. British Journal of Educational Studies, 55(4), 466-468. https://doi.org/10.1111/j.1467-8527.2007.00388_2.x

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from Block-Based to Text-Based Programming Languages. Proceedings - 2018 6th International Conference on Learning and Teaching in Computing and Engineering, LaTiCE 2018, (April 2018), 57-64. https://doi.org/10.1109/LaTICE.2018.000-5

Murcia, K., Pepper, C., Joubert, M., Cross, E., & Wilson, S. (2020). A framework for identifying and developing children's creative thinking while coding with digital technologies. Issues in Educational Research, 30(4), 1395-1417.

Olabe, J. C., Olabe, M. A., Basogain, X., Maiz, I., & Castaño, C. (2011). Programming and Robotics with Scratch in Primary Education. Education in a Technological World: Communicating Current and Emerging Research and Technological Efforts, (July 2016), 356-363.

Owen-Hill, A. (2021). What Are the Different Programming Methods for Robots? https://blog.robotiq.com

Papadakis, S. (2020). Robots and Robotics Kits for Early Childhood and First School Age. International Journal of Interactive Mobile Technologies, 14(18), 34–56. https://doi.org/10.3991/IJIM.V14I18.16631

Papadakis, S., & Kalogiannakis, M. (2022). Learning Computational Thinking Development in Young Children With Bee-Bot Educational Robotics. Research Anthology on Computational Thinking, Programming, and Robotics in the classroom, 2, 926-947. https://doi.org/10.4018/978-1-6684-2411-7.CH040

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). Using Scratch and App Inventor for teaching introductory programming in secondary education. A case study. International Journal of Technology Enhanced Learning, 8(3-4), 217-233. https://doi.org/10.1504/IJTEL.2016.082317

Papadakis, S. (2020). Tools for evaluating educational apps for young children: a systematic review of the literature. Interactive Technology and Smart Education, 18(1), 18-49.

Papadakis, S. (2022). Apps to promote computational thinking concepts and coding skills in children of preschool and pre-primary school age. In Research Anthology on Computational Thinking, Programming, and Robotics in the Classroom (pp. 610-630). IGI Global.

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating the effectiveness of a game-based learning approach in modifying students' behavioural outcomes and competence, in an introductory programming course. A case study in Greece. International Journal of Teaching and Case Studies, 10(3), 235-250.

Papadakis, S., & Orfanakis, V. (2018). Comparing novice programing environments for use in secondary education: App Inventor for Android vs. Alice. International Journal of Technology Enhanced Learning, 10(1-2), 44-72.

Papadakis, S., Vaiopoulou, J., Sifaki, E., Stamovlasis, D., & Kalogiannakis, M. (2021). Attitudes towards the use of educational robotics: Exploring pre-service and in-service early childhood teacher profiles. Education Sciences, 11(5), 204.

Papadakis, S.; Trampas, A.; Barianos, A.; Kalogiannakis, M. and Vidakis, N. (2020). Evaluating the Learning Process: The "ThimelEdu" Educational Game Case Study. In Proceedings of the 12th International Conference on Computer Supported Education - Volume 2: CSEDU, ISBN 978-989-758-417-6, 290-298.
https://doi.org/10.5220/0009379902900298

Plaza, P., Castro, M., Merino, J., Restivo, T., Peixoto, A., Gonzalez, C., & Strachan, R. (2020). Educational Robotics for All: Gender, Diversity, and Inclusion in STEAM. Proceedings of 2020 IEEE Learning With MOOCS, LWMOOCS 2020, 19-24.
https://doi.org/10.1109/LWMOOCS50143.2020.9234372

Plaza, P., Sancristobal, E., Carro, G., Castro, M., & Blazquez, M. (2018). Scratch day to introduce robotics. IEEE Global Engineering Education Conference, EDUCON, 2018-April, 208-216.
https://doi.org/10.1109/EDUCON.2018.8363230

Psycharis, S. (2018). Computational Thinking, Engineering Epistemology and STEM Epistemology: A Primary Approach to Computational Pedagogy. Advances in Intelligent Systems and Computing, 917, 689-698.
https://doi.org/10.1007/978-3-030-11935-5_65

Raspberry, P. (n.d.). Projects, Computer coding for kids and teens, Raspberry Pi.
https://projects.raspberrypi.org

Ruzzenente, M., Koo, M., Nielsen, K., Grespan, L., & Fiorini, P. (2012). A Review of Robotics Kits for Tertiary Education. Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum, 153-162.

Tzagkaraki, E., Papadakis, S., & Kalogiannakis, M. (2021). Exploring the Use of Educational Robotics in primary school and its possible place in the curricula. In Educational Robotics International Conference (pp. 216-229). Springer, Cham.

Üşengül, L., & Bahçeci, F. (2020). The Effect of Lego Wedo 2.0 Education on Academic Achievement and Attitudes and Computational Thinking Skills of Learners toward Science. World Journal of Education, 10(4), 83.
https://doi.org/10.5430/WJE.V10N4P83

Vega, J. (2018). Educational Framework Using Robots with Vision for Constructivist Teaching Robotics to Pre-university Students, Universidad de Alicante, 1-208.
http://hdl.handle.net/10045/90414

Verawati, A., Agustito, D., Pusporini, W., Utami, W., & Widodo, S. (2022). Designing Android learning media to improve problem-solving skills of ratio. Advances in Mobile Learning Educational Research, 2(1), 216-224.
https://doi.org/10.25082/AMLER.2022.01.005

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. Computers and Education, 142, 103646.
https://doi.org/10.1016/j.compedu.2019.103646

Weintrop, David, & Holbert, N. (2017). From Blocks to Text and Back: Programming Patterns in a Dual-Modality Environment. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, March 2017, 633-638.
https://doi.org/10.1145/3017680.3017707

Weintrop, David, & Wilensky, U. (2017). Between a block and a typeface: Designing and evaluating hybrid programming environments. IDC 2017 - Proceedings of the 2017 ACM Conference on Interaction Design and Children, 183-192.
https://doi.org/10.1145/3078072.3079715

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.
https://doi.org/10.1098/RSTA.2008.0118

Yakman, G. (2008). STEAM Education: an overview of creating a model of integrative education. PATT.

Zaher, A. A., & Hussain, G. A. (2020). STEAM-based active learning approach to selected topics in electrical/computer engineering. IEEE Global Engineering Education Conference, EDUCON, April 2020, 1752-1757.
https://doi.org/10.1109/EDUCON45650.2020.9125367