

Visual Programming for Artificial Intelligent and Robotic Application (VPAR) Framework

Goragod PONGTHANISORN ^a, Waranrach VIRIYAVIT ^{b,c}, Thatsanee CHAROENPORN ^d, and Virach SORNLERLTLAMVANICH ^{d,e,1}

^a*Department of Computer Engineering, Faculty of Engineering, Thai-Nichi Institute of Technology, Thailand.*

^b*School of ICT, Sirindhorn International Institute of Technology, Thammasat University, Thailand.*

^c*Graduate School of Science and Engineering, Chiba University, Japan.*

^d*Asia AI Institute (AII), Faculty of Data Science, Musashino University, Japan.*

^e*Faculty of Engineering, Thammasat University, Thailand.*

Abstract. Computer programming is popularized in 21st century education in terms of allowing intensive logical thinking for students. Artificial Intelligent and robotic field is considered to be the most attractive for programming today. However, for the first-time learners and novice programmers, they may encounter a difficulty in understanding the text-based style programming language with its special syntax, semantic, libraries, and the structure of the program itself. In this work, we proposed a visual programming environment for artificial intelligent and robotic application using Google Blockly. The development framework is a web application which is capable of using Google Blockly to create a program and translate the result of visual programming style to conventional text-based programming. This allows almost instant programming capability for learners of programming in such a complex system.

Keywords. Artificial Intelligent, Robotic, Web Application, Visual Programming Framework, Block-based Programming, Blockly

1. Introduction

Since the starting of human history, communication is an effective tool which leads to emerging of ancient civilization and a foundation of the modern society. The purpose of communication is a knowledge sharing in many disciplines by utilizing a verbal communication which is comprehensible by each other. However, a verbal communication has a flaw in which information may be altered before reaching a destination. To preserve the correctness, a writing system was invented where a verbal expression is described in a formal set of characters creating a pronunciation guide for a

¹ Virach Sornlertlamvanich, Asia AI Institute (AII), Faculty of Data Science, Musashino University, 3-3-3 Ariake Koto-ku, Tokyo, 135-8181, Japan; and Faculty of Engineering, Thammasat University, 99 Moo 18, Paholyothin Road, Klong Nueng, Klong Luang, Pathumthani 12120, Thailand; E-mail: virach@gmail.com

reader. These inventions have been used since then. The writing system has made a great contribution to many innovations while the knowledge transfer is continuously improved. Later, human has transcended in a way of communication, from just only to themselves, to a silicon-based device (i.e., computer). The language of computer, called machine code, is a series of binary numbers which represents the instructions that the computer have to execute accordingly. In early age of programming, the programmers are used to communicate in such an almost-incomprehensible language to create a computer program. Nonetheless, the primitive way of the communication with computer obstructs higher feature-rich applications. This leads to a birth of programming language.

Rather than writing a computer program directly in machine code, a programming language compromises human language and the machine language. Programming language is a combination of a language alphabets (mainly in English) and special characters to create instructions. These instructions, later, are going to be compiled by a compiler, a programming language to machine code translator, to create a computer program. The development of programming language allows more complex and newer features to be programmed by an aid of provided tools that can be used by programmers. Nowadays, there is almost 600 programming languages for each type of application ranging from a computer hardware interfacing to a cloud computing and a web-service.

Currently, Python is one of the most popular programming languages [1], which its application lies from an education to research area. Not only its simplicity, but also its community where various modules and libraries for applications are shared. As in the field of Artificial Intelligence, which an intensive computation is required, Python provides a framework for ease of programming in complex mathematic related field, for instance, Keras and Pandas. While in robotic applications, conventional programming languages (e.g., C/C++) are widely used because its simplicity and low-level feather for hardware related programming. However, the recent emergence of MicroPython has caused a trend of microcontroller application to be written in Python. This leads to rapid prototyping and being more user-friendly for a first-time learner as the Python syntax is simple. Including its Object-oriented paradigm, the new era of microcontroller is opening.

Although the design of Python is simple and introduced in object-oriented style, just like other programming languages, it is still problematic from being a text-based language. Firstly, a text-based programming contains a considerably amount of mathematic sign to express its instruction. This reflects the original purpose of programming, to perform a complex mathematic computation. While application of programming in modern day has covered wide range of field already. Secondly, as born from mathematic, programmer must pose an ability to transform real-world problem into a logical and instruction representation in programming. Not to mention the commonly found mistake for a novice, a syntax violation. Programming languages must strictly follow the syntax otherwise execution of a program is not initiated.

The complexity of text-based programming language may hinder learners' interest in programming [2]. When novice programmers try to write their first program, the first lesson is always a simple program, such as a print function (the most empirical output from their program). This is a process of making a learner get familiar with a style (syntax) and specific function provided by each programming language. However, the naiver, the more mistakes are likely to be found, e.g., missing terminating character (; semi-colon) in C language, style of writing a conditional statement, and scope of for-loop operation defined by curly brace. Not only syntax, but novice programmers may also encounter difficulty in trying to evaluate a logical error. A logical error is also a common mistake, no exception to most experience programmers, and is not easy to

evaluate since a logical error requires some moderate competence in a programming including a functionality of currently used the programming language. While the core function of programming language is to solve a problem by creating a set of instructions so called algorithm, a programmer tends to waste time for this process. These difficulties obstruct the main purpose of programming.

Recently, visual programming has become popular for the novice and first-time learner. This paradigm of programming, instead of text, uses a block or another notation to represent a logical flow. It is also called a block-based programming style. This enables more intuitive of a computer programming. Moreover, a graphical representation of visual programming evaluates information in the closest manner to human mental representation of real-world problems [3]. There are multiple well-known visual programming styles in broad range, for example a model-based design of MATLAB [4] which represents an equation in a block and flow of logic using a flow-based design. LabView is used for an embedded application that implements a graphic of an electronic device and sign for the representation the system [5], Scratch, MIT Block and Google Blockly [6, 7, 8] which employ a concept of representing computer instruction into a block called block-based programming. Visual programming seems promising for a new programming paradigm as multiple applications employed the idea and concept. For instance, the works of [9] and [10] implemented a visual programming for a machine learning application through a web application. Especially, in [10], the broad of application using MIT block is introduced. [11, 12, 13, 14, 15] have selected a Google Blockly, an open-source block-based programming which are developed on web application as a tool for a visual programming and apply to a variety of application ranging from robot to Augment Reality (AR) application. However, [11, 12, 13, 14, 15] have some limitations. These works require a user to install and setup a required tool before. Although [9] and [10] are accessible via online using a general web browser, the work of [10] which can use as robot programmer, requires an installation of external tool (e.g., mLink before connecting to a physical robot). The researchers of this present study found the flaw of the robotic application on such a feather-rich framework and saw its potential for AI application. Thus, we have designed and developed the visual programming framework which is able to develop AI and robotic application without any additional tool installation. For a visual programming and translation of text-based, we have selected Google Blockly and Python.

Table 1. Comparison of Robot for Programming [15]

| Product | Processor | Processor Clock Speed (MHz) | Supported Programming Language |
|-----------------|--------------------------------------|-----------------------------|---|
| Mindstorms EV3 | ARM 92EJ-S (32-bit) | 300 | <ul style="list-style-type: none"> • The EV3 on Brick Programming |
| Robot-PICA | PIC16F887 (8-bit) | 8 | <ul style="list-style-type: none"> • Assembly • BASIC • C Language |
| Scribbler3 | Multi-core Propellor P8X32A (32-bit) | 80 | <ul style="list-style-type: none"> • C Language • Blockly • Scribbler S3 GUI |
| Robot-CreatorXT | ATmega644P (8-bit) | 16 | <ul style="list-style-type: none"> • C/C++ Language |
| IPST-MicroBOX | ATmega644P (8-bit) | 16 | <ul style="list-style-type: none"> • C/C++ Language |
| mBot1.1 | ATmega328 (8-bit) | 16 | <ul style="list-style-type: none"> • Scratch 2.0 • C/C++ Language |

As for a robotic application, the main application is to retrieve information from robot sensory system and evaluate a robot action. There are many robot programming frameworks currently placed in the education market. Table 1. provides information of well-known programming robot products from various manufacturers. Three key attributes are used to describe and distinguish the robot system.

Information in Table 1. implies that a robot programming mostly supports text-based programming language. Although some of them (Mindstorm3, Scribbler3 and mBot1.1) employ a visual programming concept to extend a range of suitable age for playing. Nonetheless, all mentioned robots are not able to perform a heavy computation application (e.g., image processing) since their performance is limited by equipped processor. These robots may categorize in application-specific robot which a range of applicable tasks are limited while the need of a general-purpose robot is rising. The term of general-purposed robot refers to a robot which is capable of execute general trivial task similar to human. For instance, a trivial task for human, object detection. Human brain and our visual sensory system are effective enough and allow us to distinguish an object shape and color then correctly describe them. Yet, for a robot, it is considered one of toughest tasks. To perform an object detection in logic computation, first, the robot needs to obtain an image (which is normally represented in multi-dimensional array). Once image is obtained, robot executes a heavy computation using high-level mathematic to extract a necessary piece of information from the image. Then, the robot executes another heavy computation to identify an object. With all said, it is impossible to execute that task in such a low performance processor on the mentioned robot.

In order to archive such a complex system, a multiple component of computation software and an effective sensory system equipped including high-performance processor to handle a heavy computation. Thus, this work selects a Temi to implement the proposed system. Temi is a personal assistant robot which embedded with high performance ARM HEXA core processor and multiple sensory system (i.e., LiDAR, microphone, 2 RGB depth camera, 5 proximity sensor, 6 time of flight sensors and IMU sensor) [16]. Equipped with such a high-performance processor, Temi robot is capable of executing a complex task in a real-world environment, for example, a navigation through a dynamical environment, an interaction via speech and conversation context awareness with the built-in NLP (Natural Language Processing) unit, a trajectory free path planning using an image and point cloud data from LiDAR.

Temi robot programming is different from typical other robot programming in term of a software structure. The robots, mentioned in table 1., is equipped with low-end processor that easily to program in a low-level language (C/C++, assembly). Not for Temi robot that equipped with high-performance processor ARM HEXA core. Since complexity of low-level language for non-embedded-field-programmer hinders a full potential, Temi robot's manufacturer decide to provide SDK (software development kit) for develop an application for Temi robot. The SDK is developed on Android framework using both Java and Kotlin language [17]. Temi SDK allows a programmer to create robot application called *skill*.

While the provided SDK soothes a difficulty of implementation of a software component directly into the robot's hardware by providing an inherited method as a callback for each event. For instance, when the robot starts speaking, the information regarding to a state of the speech can be obtained from derived method call *onTtsListener*. Although program implemented by the provided SDK is easy, the program structure is becoming more complicate. Since multiple components are required in single application which is commonly found. The more codes and processes for a program, the harder

program evaluation becomes. A visual programming paradigm should help mitigating the matter. Rather than describe the robot response of behavior in text-based programming, visual-based block seems far easier for novice.

In addition, the robot requires a multiple step of a configuration on Android Studio and ADB (Android Debug Bridge) in order to perform an installation of the developed program. This task's complexity is comparable to understanding of the robot's program flow.

In this work, not only AI application but also such the feature-rich robot, Temi is going to be programmed by visual programming Google Blockly on an online IDE which requires minimal configuration. This enables even a novice programmer to create a simple application for Temi. The detail of implementation of both AI and robotic application will be discussed later.

The paper is organized in the followings. Section 2 outlines the proposal of a framework for AI and robotic application development using visual programming style. In this section, we introduce a motivation and design concept of the system and describe an overview architecture including the flow of the system. Section 3 discusses the implementation of block-based concept into a simple Neural Network (NN) programming. The section presents a basic parameter of NN, mathematic and block representation. Section 4 discusses a concept designed for programming Temi, personal assistant robot via visual programming language. We implement an additional component to the proposed system including Temi application environment allowing such a new paradigm of program to the robot. After the key idea of design and implementation is introduced, we include a prototype of web-application of the proposed framework with functions like an IDE which resides remotely on server. In Section 5, we discuss the limitations of our proposed system. Lastly, a conclusion is drawn, and some on-going and planned work is shown.

2. Proposal of Visual Programming for AI and Robotic Application via Online IDE

In order to create a computer program, an integrated development environment (IDE) is required. There are two basic components for every IDE, 1) text editor and 2) compiler or interpreter. Text editor is used for a text input to create a programming instruction. After the program is written, a compiler or interpreter (depending on which type of programming language) translates the human comprehensible language into a hardware native machine code. One of the drawbacks is when starting learning a program, programmers do not only learn syntax and semantic of the language but also have to learn how to perform an installation and configurations of the additional tools. Luckily, many modern IDEs have included all the required components in a single installation package. Yet, another issue is raised, with a different environment of each computer (e.g., OS, version, and previously install program). These can affect the operation of an IDE. In order to avoid this matter but still provide programming environment, a web-application online IDE seems to be a promising solution. A general idea of online-IDE is to provide a programming service via a web-browser without any installation of specified tool set. Moreover, variant of environment, compared to locally installed IDE, is relatively small (concerning on web-browser and its version).

The proposed visual programming for AI and Robotic Application system is designed for beginners in the world of programming. The key concept of the proposed system is the visual programming for representing a complex computation and logic flow

while providing an insight of a program by translating the visual to text-based language. Users should be able to visualize a program flow via visual-based approach. Once they are competent in the flow of logic, a text-based approach is introduced to them. Thus, our system uses block-based for visualizing of program and providing a translation of that block-based to text-based code. In addition, the proposed system was designed to be web-application. The concern is on the complexity of tool setup and installation when writing a program. In that sense, the proposed system should be designed in ready-to-use manner without further installation.

As for block and text-based programming language, Python is selected because of its simplicity and suitability for novice programmers. For block-based programming, we use Google Blockly. Google Blockly is developed using JavaScript and executed on a client-side web-browser. Google Blockly has built-in translation unit from block-based to text-based conversion (JavaScript, Python, Lua, Dart and PHP) [8].

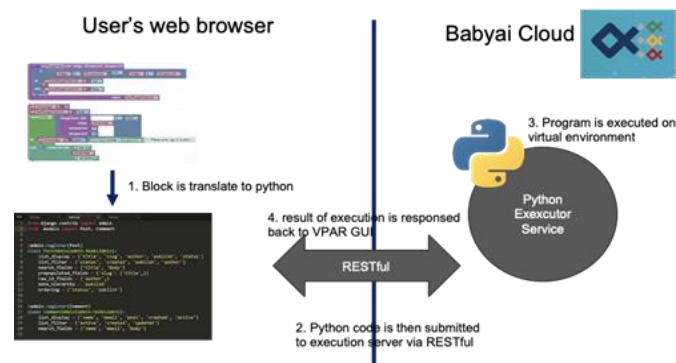


Figure 1. The Architecture of the proposed Visual Programming for AI and Robotic Application system.

The proposed system is comprised of web-application which is developed by React framework. Then the developed web-application is executed on the web browser of client side. The key role of the web-application is to provide user-friendly GUI to users, block code development using Blockly framework and a translation of block-base programming to Python programming language. Nonetheless, a typical web-browser poses strict security protocol which an arbitrary script execution is prohibited. The solution is that the translated program should be executed somewhere else. This leads to the second component of the system, Python Execution Server (PES). PES is a web-service hosted on cloud and exposed an available service through HTTP REST API. The primary features of the system are to 1) execute a program code, 2) export the current workspace, and 3) send a result of execution by HTTP response. PES is considered as a backbone of our proposed system. PES executes Python code in docker container and virtual environment so that any harmful operation will not directly affect the server itself. PES is used by both AI and robotic application, despite variant on module and library usage. PES is able to retrieve necessary package through Python package installer, *pip*, before execution. In addition, PES also supports multiple language input and display, as mandatory to higher programming language as well. As for web-application, this feature may be provided by web-browser, but there is not on the server script, such as PES. However, the translated program must be executed by PES. Thus, it is equipped with

Unicode character encode/decode service to avoid incorrect output when dealing with Unicode character.

The proposed system work process starts with a block-based program which is translated into a text-based programming language result a code in Python. The translated program is then transferred to Python Execution Server (PES) via REST API. Once the code is transferred, PES determines the code program and executes the following code. The result of execution is then retrieved and sent back via the response of HTTP request to client. The communication between web-application and PES is on REST API fashion which PES exposes only necessary commands for security purpose.

3. A Design of Block-based Programming for AI Application Case Study: Neural Network

One of the purposes of the proposed system is to create Artificial Intelligent (AI) application using Blockly. With a broad field of AI application, the proposed system aims for the fundamental of AI application, Neural Network (NN), which is a great starter for understanding. The principle of NN starts with a mimic of human perceptron which connects and creates a neural network by mathematic equation. The mathematic model derived from NN is illustrated by the simple model in Figure 2.

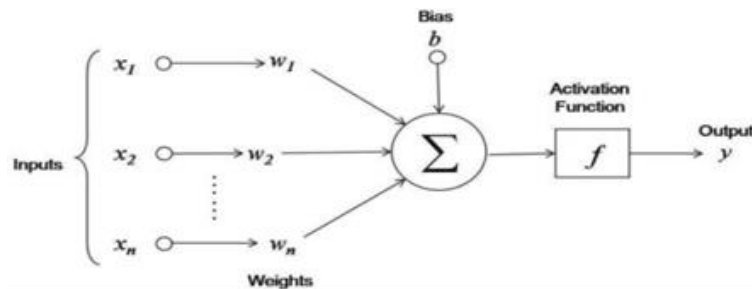


Figure 2. Single Perceptron Neural Model.

For a representation of neural to mathematics and computer programming language together with an activate function for an output mapping, the equation of the system is described in (1) as following.

$$y = f(\sum_1^n(x_n * w_n + b)) \quad (1)$$

For novices or first-time learners who are not familiar with programming, realization of a model and a program flow design is not easy tasks. Lack of understanding is potentially a great obstacle for further implementation or modification. An alternative way for providing better understanding of application is visual programming. The proposed system compromises an ease of understanding of model view and sematic of programming language by representation of neural unit into a block command.

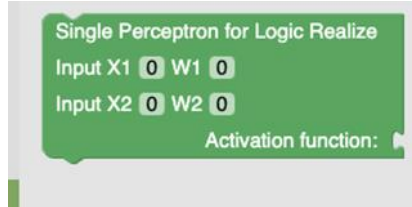


Figure 3. Block Representation of Single Perceptron.

For case study, the logic realization application of single perceptron (2 inputs and 1 output) is implemented. The block representation of a perceptron is illustrated in Figure 3.

As shown in equation (1), a neural network requires an activation function for an output mapping. Plug-and-play design principle is used to provide a wide range of experiment to examine the output of NN due to activation a certain function. Thus, the representation of activation function to block is designed and shown in Figure 4.



Figure 4. Block Representation of Activation Function.

The activation function block connector is different comparing with NN block since they are designed to connect to a NN block only. A parameters of activation block are projected by their equation of (2) and (3) as follows.

$$f(x) \begin{cases} 0 & \text{for } x < Zeta \\ 1 & \text{for } x \geq Zeta \end{cases} \quad (2)$$

$$f(x) = \frac{1}{1+e^{(zeta-alpha)}} \quad (3)$$

Equation (2) is called binary step function which the output is determined by Zeta to be 0 or 1 (integer), while equation (3) describes a sigmoid function which the output is mapped between 0 and 1 (floating point).

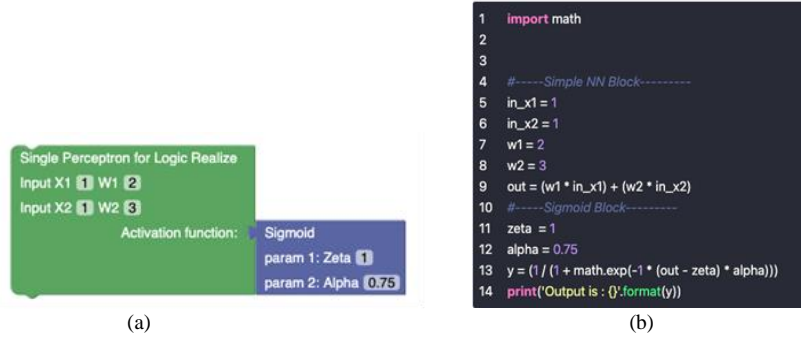


Figure 5. (a) Block Program (b) Translated Code

Instead of writing a NN program, the block code of NN provided by the proposed system is used. Figure 5 (a) and (b) demonstrates a NN application for logic realization using block program including translated code.

Not only a block-styled code, but an user can also start learning a concept of programming with Python by examining a translated code as well. As a plug-and-play design, an activation function can be replaced easily by drag-and-drop manner. The translated code and output are then changed due to the difference of activation function.

4. A Design of Blockly for Personal Assistant Robot (Temi)

Temi is a personal assistance robot, equipped with multiple useful functionalities, e.g., user-interaction via built-in natural language processing (speech to text and text to speech), a camera which enables facial recognition and person tracking, and internet connectivity for android applications such as YouTube and simple web-browser. The robot has a set of functions called skills which describe abilities of robot for certain tasks. Table 2 provides a detailed description of the robot abilities.

Table 2. Summary of the Robot Skills [16].

| Skill | Description |
|-------------------------------|---|
| Location and Map | Location saving, Map generation, and start-to-location navigation |
| Movement | Control direction of robot movement |
| Speech | Temi-user interaction via speech and simple conversation |
| User and Telepresence | Make a video calling using saved contact via application called Temi App |
| Face Recognition and Sequence | Face detection unit using built-in RGB camera |
| Detection and Interaction | Exploit a usefulness of Face Recognition ability to create a human detection |
| Follow | Exploit a usefulness of Face Recognition and Detection ability, enabling a follow-trackable-person navigation |

Although a preliminary installed program can easily suffice a daily life usage, Temi developer provides an SDK for programmers to create their own application (called skill) as well. The provided SDK is based on Android and Kotlin which provide an API access to a variety of robot features [16].

The proposed framework aims to provide ease of programming framework of Temi robot for novice programmers by projection of Temi skill to a block-based programming,

Blockly. The block-based program is then translated into Python and transferred to execute server where the command is submitted to the robot through MQTT communication. The MQTT communication does not provide only the access to the robot for command issuing, but also robot status, e.g., battery level and current skill status in order to provide more insight information of robot on translated program. The overall operation of translated program and the robot communication is described in Figure 6.

The visual programming for Temi robot consists of multiple components which enable a feature and provide an alternative of the robot's program execution. The components are listed as follow

- VPAR, a web-application for a Blockly-to-Python translation
- PES, a code execution service
- MQTT broker, a mediator between the Python code and the robot
- The developed android application that bridges (using MQTT) between the Python code and the robot

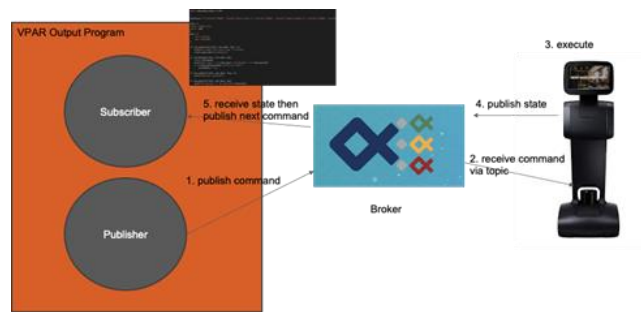


Figure 6. Overall Operation of Translated Robot Program Execution

In order to program the robot, the proposed system provides a block-based program which can be created using the web framework of proposed system. The robot skills are represented by a block type. These blocks can be connected to each other to create an application for the robot. The available block command is shown in Figure 7(a) and 7(b).

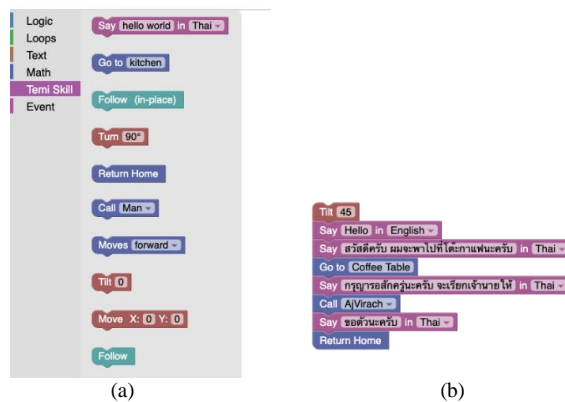



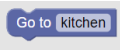

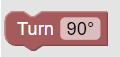
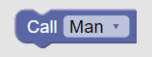

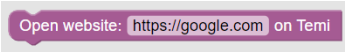
Figure 7. (a) Block Command for the Robot (b) Example Program

Figure 7 (a) shows a set of commands which can be used for creating a program for the robot while Figure 7 (b) demonstrates a program from blocks. Table 3 provides information of robot skill represented by blocks as follows.

The proposed system projects a skill set of the robot in a block representation which each block requires different parameter. For example, *Say* command requires two parameters, the text to be spoken by the robot and language option according to the previous option. Each block can be connected, and the program will execute a command accordingly. The translated program controls a timing of command transferred to the robot. If previous command has not yet finished, the next command must wait to create a sequence of block execution.

To achieve such a feature and communication of the proposed system, the environment for the robot programming comprises two essential units.

Table 3. Skill and Block Representation

| Block | Skill | Description | Parameter |
|---|--------------|--|-------------------------------|
|  | Speak | Read out a text in the box, language must be agreed | Text, Language Option |
|  | Go to | Tell the robot to go to saved location | Text (in all lower case) |
|  | Track | Track a person but the robot will not move along with tracked target | - |
|  | Turn | Robot turn itself specified by degree | - |
|  | Call | Robot makes a video call to a person specified in dropdown menu | Selected person from dropdown |
|  | Tilt head | Move head along its moving axis (between 37 – 53) | Degree 37 - 53 |
|  | Open website | Temu open the specific URL in built-in web browser | |

4.1. Temi Command Actuator Android Application

The Android application for MQTT communication and the robot instruction decoder are installed into the robot, allowing a user to experience ready-to-use application development framework.

Table 4. API Overview for Location Related [17]

| Return | Method | Description |
|--------|-------------------------------|--|
| Void | speak (TtsRequest ttsRequest) | Ask Temi to speak (play TTS) |
| Void | cancelAllTtsRequests | cancel TTS request |
| Void | wakeup() | Wake Temi up |
| String | getWakeupWord() | Get current wake-up word |
| Void | askQuestion(String question) | Temi speaks actively and waits for user reply |
| Void | finishConversation() | Finish a conversation (Stop recording for ASR) |
| Void | startDefaultNlu(String text) | Trigger default NLU service |

The key functions of the application are 1) To bridge the Python program from VPAR and the robot 2) To translate instruction from JSON format into a function call accordingly to command 3) To issue a command received from the Python program to the robot's SDK. These works implement the application using provided SDK on Java programming language which contains an API for accessing Temi skill. Table 4 shows some API for Temi robot. The application is simple and provide only information for debugging purpose (at this state of development. Most of the application operation is to perform MQTT connection, to receive JSON package to an instruction conversion and to control the robot's action flow. Figure 10 shows some of an essential function inside the bridge application *i.e.*: MQTT communication and an instruction (in JSON format) decoder function.

```

/*----- ACTION DECODER -----*/
public void actionDecoder(JSONObject actionInfo) {
    try {
        switch (actionInfo.get("action").toString()) {
            case "speak":
                speak(actionInfo.get("content").toString());
                if(actionInfo.get("language").equals("zh")) {
                    RobotSpeak_In(actionInfo.get("content").toString());
                }
                else if(actionInfo.get("language").equals("en")) {
                    speak(actionInfo.get("content").toString());
                }
                break;
            case "goto":
                if(goto(actionInfo.get("content").toString())) {
                    break;
                }
            case "call":
                printLog("Call to " + actionInfo.get("content").toString());
                startPresence("blockly-agent", actionInfo.get("content").toString());
                break;
            case "openweb":
                String url = actionInfo.get("content").toString();
                Intent webIntent = new Intent(getApplicationContext(), WebViewActivity.class);
                webIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                webIntent.putExtra(WEBVIEW_URL, actionInfo.get("content").toString());
                try {
                    getApplicationContext().startActivity(webIntent);
                } catch (ActivityNotFoundException e) {
                    e.printStackTrace();
                }
                break;
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

/*----- MQTT -----*/
private void startMQTTClientInterface() {
    String cmd_topic = "temi/" + temi_serial + "/temi-cmd";
    printLog(cmd_topic);

    mqttCmdInterface = new MqttWrapper(getApplicationContext(), cmd_topic, "VirachTemiRobotClient");
    mqttCmdInterface.setCallback(new MqttCallbackExtended() {
        @Override
        public void connectComplete(boolean reconnect, String serverURI) {
            printLog("Connected to " + serverURI.toString());
        }

        @Override
        public void connectionLost(Throwable cause) {
            mqttCmdInterface.reconnect();
        }

        @Override
        public void messageArrived(String topic, MqttMessage message) throws Exception {
            printLog("From topic: " + topic.toString());
            printLog("Arrived message: " + message.toString());
            try {
                JSONObject jsonObject = new JSONObject(message.toString());
                actionDecoder(jsonObject);
            } catch (JSONException e) {
                Log.d("Error", e.toString());
            }
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken token) {
        }
    });
}

```

Figure 10. Partial of the Bridge Application Java Code

When a command is issued by the program, it sends a requested command to robot internal system enabling non-blocking program paradigm. The internal system, then, raises a status associated with issued command to identify the current status via callback API. Table 5. shows some of callback API for built-in text-to-speech system, and Figure 11. shows pseudo code for overall application operation.

Table 5. Speech-related Callback [17]

| Interface | Description |
|--|---|
| TtsListener | TTS status listener |
| WakeupWordListener | Wake-up event listener |
| AsrListener | ASR result listener |
| ConversationViewAttachesListener | Conversation view attaches listener |
| OnTtsVisualizerWaveFormDataChangedListener | Listener for wave form data changes of TTS audio visualizer |
| OnTtsVisualizerFftDataChangedListener | Listener for FFT data changes of TTS audio visualizer |

```
procedure: application(robot, mqtt, robot_status_callback,)
  robot ← RobotInstance()
  mqtt ← MQTTInstance()
  arrived_message ← Null
  loop forever :
    if arrive_message is not Null
      action ← decode(arrive_message)
      robot_execute(action):
        while status is not done:
          if status is not prev_status:
            mqtt_publish(status)
          end if
        end while
    end if
  end loop
```

Figure 11. Pseudo Code for Overall Application Operation.

The proposed system has exploited this programming paradigm, together with MQTT communication. Then, the robot skill and status can be realized through MQTT communication.

4.2. Robot module for Translated Program

Python execution server is a shared component between AI and robotic Python applications which are the result from block code translated into Python. Although an AI application may implement a built-in module or site package module (installed by pip), there is no module that provides an ability to control the robot. Thus, the proposed system for robotic application has integrated a special Python module for interacting with the robot via MQTT. As shown in Figure 12, the translated program for the robot controlling and block code are described.

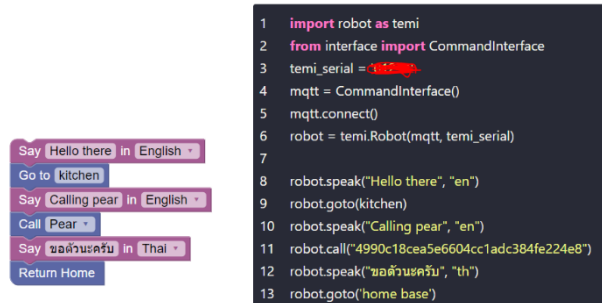


Figure 12. Block Program VS Translated Python Code from Block Program

The robot module is comprised of two units which are tightly coupling to each other. The first module, named *robot*, is written in Python. The main role of the module is to converse an issue command into predefined MQTT package for the robot in JSON format. As for delivery of the constructed message, including the robot status retrieval, the command is handled by a module named *interface*. This module is tightly coupling with robot module which creates a message and manages a program flow according to the robot status. Another role of this module is to program the robot directly with Python code.

5. Results and Limitations

The proposed system is deployed on a web-browser as web application using React framework. GUI of the web-application is shown in Figure 13. The web application has two variants. The first one is for a robotic application, and the second one is for AI application with the same GUI. The main difference is the provided blocks in the web application. The components of web application are described by numbering labels as follows:

- 1 - A block menu for desire block selection
- 2 - Canvas where the blocks program is created.
- 3 - Output result of block program
- 4 - Preview Python code for learning
- 5 - Control Menu

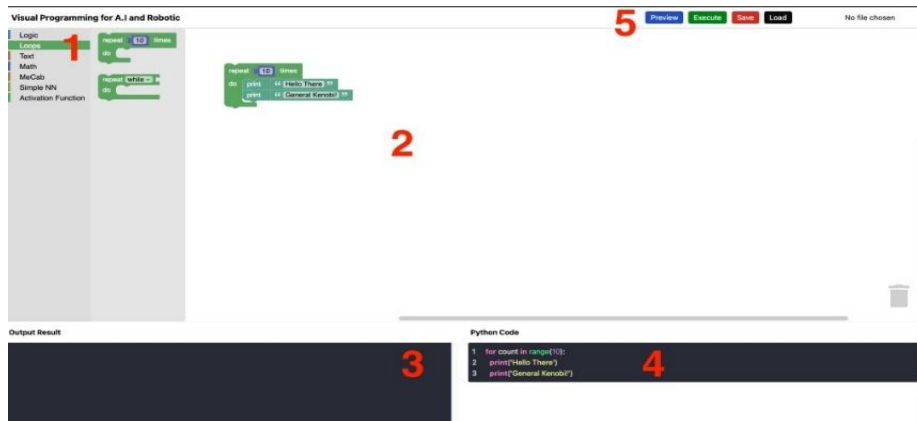


Figure 13. VPAR Web Application GUI.

The web application is considered as online-IDE for programming which provides editor, output result, and remote compiler. The block program is translated into Python code and then executed remotely. Once the execution is completed, the result is transferred back to be displayed on the web application. The web application also provides a save/load menu in which the users can import an existing program in Blockly XML file format to the web. The save button provides an export context to the web where the created block program can be saved and downloaded to a local computer for further usage.

As for the robot application, Temi, it's required to pre-install the developed bridge program to enable MQTT communication. The MQTT connection is used by the Python code (output from VPAR) which is executed by PES on cloud. Once the execution is finished, PES returns a result and displays on the web-application. The bridge application GUI, as shown in Figure 13, provides information of currently executing action of the Python code. There are two buttons for cancelling a current executing action and stopping MQTT communication. In other word, the robot doesn't receive any command at all



Figure 14. (a) The Python Code Executed on PES
(b) Bridge Application GUI

However, there are some drawbacks of the design of the proposed system. Firstly, as the translated code is compiled and executed remotely, the proposed system requires constant internet connection. There is no local application installed or the accessing of local Python interpreter for the security reason. In addition, the proposed system must finish the execution of translating program before the result can be retrieved to be displayed on the web application. It is because the proposed system relies on REST API which is basically the HTTP communication. As a result, the interpreter style of Python is not available in the proposed system as well as debugging. Moreover, the robotic application is also affected, so that the user cannot realize a communication in real-time since its execution of the program must be finished first.

6. Conclusion and Future Work

In this work, we proposed a visual programming for AI and robotic application (VPAR) framework. The system utilizes a Google Blockly for creating an AI and robotic application for novice programmers. The proposed system is fully executed remotely and provides an access to the system via the developed web application. The web application is based on React framework which serves as an online-IDE for programming. It enables save/load a workspace including previewing of translated-from-block-to-code program for better understanding. The proposed system allows a block-based usage to develop AI and robotic applications for Neural Network (NN) and Temi robot programming respectively. The result of program execution can be examined on the built-in web application as well.

For future work, we are going to provide more blocks for AI applications. We aim to provide the best experience for users by adding an interactive block-to-NN model which can dynamically generate a NN model when the block is connected. For robotic applications, we are going to introduce an event-specified action block that the robot will execute a block code according to self-defined event. This results in rapid prototype Internet of Robotic Things. As for web application, we plan to add programming mode for Python in case of more experienced users.

Acknowledgement

The authors gratefully acknowledge the financial support provided by Thammasat University Research Fund under the TSRI, Contract No. TUFF19/2564, for the project titled “AI Ready City Network in RUN”, based on the RUN Digital Cluster collaboration scheme.

References

- [1] Stephen C. Top Programming Languages. IEEE Spectrum 2020. Retrieve Jan 19, 2021, Available from <https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020>.
- [2] Kris P, Stacey E, and Leanne MH. Through the looking glass: teaching CS0 with Alice. SIGCSE Bull. 39. 2007 Mar, 40(1). pp.213–217.
- [3] Brad AM. Taxonomies of visual programming and program visualization. Journal of Visual Languages & Computing. 1990 Mar, 1(1). pp.97 – 123.

- [4] MathWorks. Simulation and Model-Based Design. Retrieve Jan 19, 2021, Available from <https://www.mathworks.com/products/simulink.html>.
- [5] Bitter R, Mohiuddin T, Nawrocki M. LabVIEW. Advanced programming techniques. 2006. Crc Press, c2007. pp.1-65.
- [6] Mitchel R, John M, Andrés MH, Natalie R, Evelyn E, Karen B, Amon M, Eric R, J, Brian S, and Yasmin K. Scratch. Programming for All. Communication of ACM. 2009 Nov; 52(11), pp. 60–67.
- [7] S.C. Pokress, and J.J.D. Veiga. MIT App Inventor Enabling Personal Mobile Computing, 2013. Retrieve Jan 19, 2021, Available from <https://arxiv.org/abs/1310.2830>.
- [8] Blockly. A JavaScript Library for Building Visual Programming Editors, Retrieve Jul 10, 2020, Available from <https://developers.google.com/blockly>.
- [9] MLBlock, Retrieve Jan 10, 2021, Available from: <https://mlblock.org/>.
- [10] MBlock, Retrieve Jan 21, 2021, Available from: <https://mblock.makeblock.com/en-us/>.
- [11] Alexandru R and Ioana C. Open Cloud Platform for Programming Embedded Systems, 2013 RoEduNet International Conference 12th Edition, Networking in Education and Research. 2013, Iasi, pp.1-5.
- [12] Nguyen VT, Jung K, Dang T. BlocklyAR: A Visual Programming Interface for Creating Augmented Reality Experiences. Electronics. 2020; 9(8):1205
- [13] David W, David CS, Patrick F and Diana F, Blockly goes to work: Block-based programming for industrial robots, 2017 IEEE Blocks and Beyond Workshop (B&B), Raleigh, NC, 2017; pp. 29-36
- [14] Nguyen, V.T., Jung, K., Dang, T. BlocklyAR: A Visual Programming Interface for Creating Augmented Reality Experiences. Electronics 2020, International Conference on Information Technology and Electrical Engineering (ICITEE), Phuket, 2017; pp. 1-6
- [15] Matenat K, Natavut K, Kamol K and Kazuhiko F: MicroPython-based educational mobile robot for computer coding learning, 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), Chonburi, 2017; pp. 1-6
- [16] Temi Personal Robot, Retrieve Nov 25, 2020, from <https://www.robotemi.com/>
- [17] Temi SDK Document Retrieve Nov 25, 2020, from <https://github.com/robotemi/sdk>