Robotics Applications Programming with ROBOTIS DREAM Systems (Excerpts)

By Chi N. Thai

CNT Robotics LLC, Buford

2019

Chapter 1: Using R+m.PLAY700™ with TASK™

Currently (2019), ROBOTIS only advertises the use of the TASK tool with the DREAM II system (http://www.robotis.us/dream/), but Thai (2018) shows that users can also interface the DREAM II Controller (CM-150) to work with the "Windows PC / MIT SCRATCH 2" environment using the ROBOTIS R+SCRATCH helper application (V. 1.09 and above - http://www.robotis.us/roboplus2/). Furthermore, in this Chapter, readers will be shown how to interface a TASK program with the Mobile App called R+m.PLAY700 which was originally created to accompany only the PLAY700 kit (i.e. Controller CM-50) (http://www.robotis.us/software/play700/). But the PLAY700 App turns out to be applicable to all Firmware 2.0 Controllers, including the CM-150 (http://emanual.robotis.com/docs/en/dxl/protocol2/) and the interfacing is done by using SMART DEVICE commands in conjunction with SMART CONSTANT parameters inside the TASK tool. The "SmartDeviceControlTable.PDF" file contains detailed information about these SMART functions and their corresponding arguments, and this document is accessible at this web link (www.cntrobotics.com/advanced-dream). This PDF file is a Chrome-translated English version of this Korean web page http://support.robotis.com/ko/software/mobile_app/rasmart/smanrt_manual.htm#Actuator_Address_0B3.

1.1 Installation and Usage of R+m.PLAY700

The latest version of R+m.PLAY700 can be downloaded and installed on your iOS or Android devices at these web links (https://play.google.com/store/apps/details?id=com.robotis.play700&hl=en or https://itunes.apple.com/us/app/play700/id1136037721?mt=8). The author had found that the Android's version of the PLAY700 App is more functional and less buggy than the iOS' version.

1.1.1 File & Folder Management Differences

For development work, the author happens to use a Windows 10 PC which interfaces well with Android devices, and all file and folder management can be done via Windows Explorer. Fig. 1.1 shows the folder/file structure for PLAY700 on an Android device.

The reader can see that a PLAY700 project has many components represented by the folder structures shown in Figs. 1.1 and 1.2.

Fig. 1.1 shows that all ROBOTIS R+m apps are installed under a main folder named "RoboPlus", off the root directory of the main storage. As one drills into the PLAY700 folder, one can see three sub-folders "Custom", "System" and "Temp". The "System" sub-folder contains the 4 example projects provided by PLAY700. The "Custom" sub-folder contains the user-initiated projects, for example the "Hello World" project where the project's standard components are stored:

Audio – for audio files to be played by the mobile device but controlled by the TASK code at run time.
Captured – where pictures captured by the PLAY700 app via the mobile device's front and back cameras are kept.
Db – where SMART databases are stored, including "Text" and "Audio Input" lists used for Speech Recognition (these items can only be modified via the editing of the Project's Tools – see Sections 1.1.2 and 1.1.3).

Image – where photos specifically used by a given project are first "registered" and then "stored" as background images (i.e. "**Bg**" sub-folder) or as foreground images (i.e. "**Fg**" sub-folder).
Motion – where MTNX (Motion) files are stored. The CM-150 is not capable of processing Motion files.
Recorded - where videos captured by the PLAY700 app via the mobile device's front and back cameras are kept.
Task – where TSKX files are stored if the R+m.TASK tool is used to create TASK codes on the mobile device.
Video – for video files to be played by the mobile device but controlled by the TASK code at run time.

The "Temp" folder is only used by the PLAY700 App for its own purposes.

For an iOS device, Fig. 1.2 shows that ROBOTIS R+m Apps are installed under a main folder named "Apps" off the root directory of the main storage. As one drills into the PLAY700 folder, one would encounter a "Documents" sub-folder which further branches out into two sub-folders "Custom" and "System" which contain similar sub-folders from Audio to Video as described in previous paragraphs.

If the user prefers to use the mobile version of the TASK tool, please consult the details in Section 1.10.

1.1.2 Settings for PLAY700 App

Fig 1.3 shows a screenshot of the main menu window for the PLAY700 App where the reader can see a "gear' icon on the top right corner. Once this "gear" icon is tapped once, the "Settings" screen is shown as in Fig. 1.4.

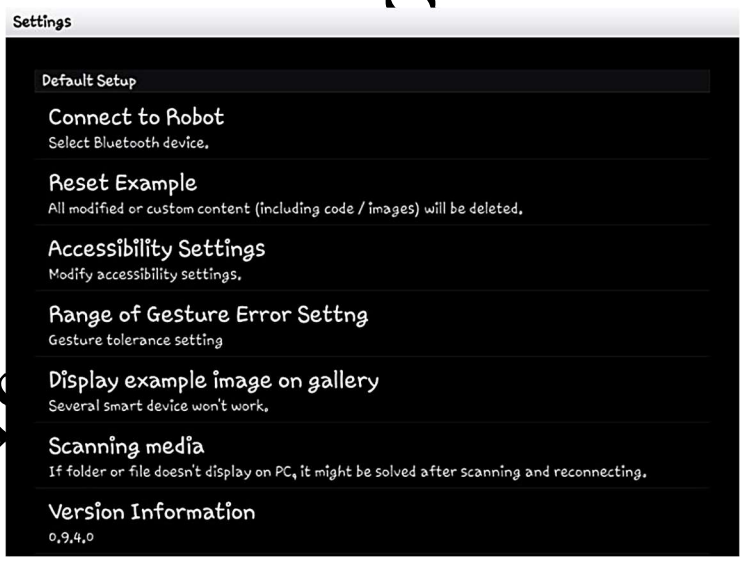

Fig. 1.4 App-level "Settings" for PLAY700 App.

The first item "Connect to Robot" allows the user to scan for new BT devices and pair them or to choose a specific BT device among those that were previously paired.

The second item "Reset example" can be used to reset all ROBOTIS example System projects to their originally installed states. This setting does not affect the Custom projects.

The third item "Accessibility Set-up" only pertains to Android devices where the user could set-reset many OS-level settings such as "Auto Rotate Screen" or "Screen Timeout". This "Accessibility" setting is not available for iOS devices.

The fourth item "Range of Gesture Error Setting" is used to set the accuracy of the gesture function (a number between 0 and 30 – available to both Android and iOS devices).

The fifth item "Display Example Image on Gallery" also pertains to Android devices only and does what its label implies.

The sixth item "Scanning Media" also pertains to Android devices only and removes the original "hidden" status of the folders and files inside the PLAY700 main folder so that Windows Explorer on a PC can "see" them.

1.1.3 Tools available for a PLAY700 Project

Going back to Fig. 1.3, when the user taps on the "Edit" button of a project, a screen as shown in Fig. 1.5 displays all the components/tools that can be used for this project. The current Android version (0.9.4.0) has all these tools operational, **but for the current iOS version (1.0.5), the following six tools are not yet functional**:

"Instrument" in the Multimedia group.
"Illumination" in the "Sensor" group.
"Received SMS", "Status Bar", "Vibration", "Application" in the "Other" group.

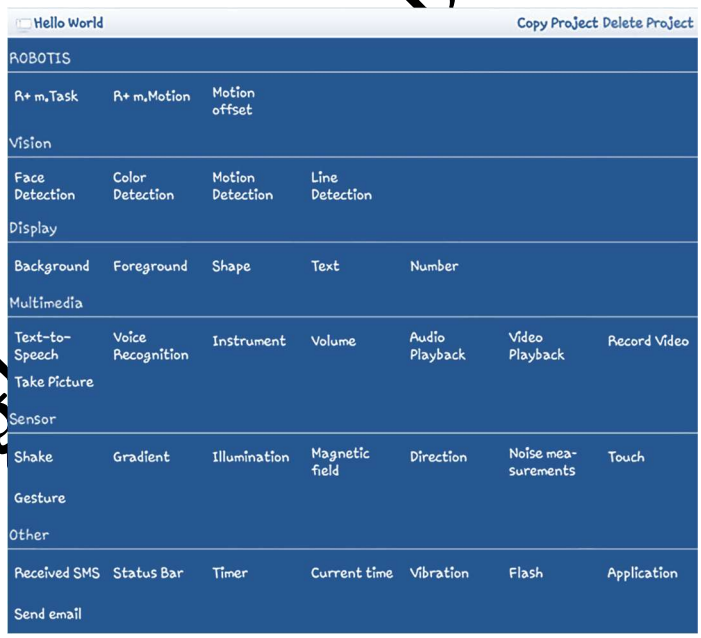

Fig. 1.5 Components/Tools of a SMART Project.

For example:

1) The "R+m.TASK" item allows the user to edit the TSKX files corresponding to this project.

2) The "Face Detection" item switches to the camera whereas the user can choose the front or back camera so that it can find a human face and then puts on a pair of sun glasses across the eyes, all in "real time".
3) The "Color Detection" item also kicks in the camera and shows a live image with a small centered region of interest where it would try to detect for 4 "colors": Black, Red, Green and Blue.
4) The "Display" section pertains to "Background" and "Foreground" images, and other items like "Text" and "Number". For example, the user can add and register Text Items if the user chooses the "Text" item in this "Display" section.
5) The "Multimedia" section can be used to record or play audio and video segments, and for text-to-speech and speech recognition tasks.
6) The "Sensor" section pertains to sensors embedded in the mobile device such as its Touch, Gesture, Gyro and Tilt sensors.
7) The "Other" section applies to "Messages" and other "Applications" that can be linked to this project.

1.2 Basic Usage of TASK with R+m.PLAY700

1.2.1 Two ways to access SMART DEVICE commands

Interestingly, the access to the SMART DEVICE commands for the CM-150 has a rather tortuous history which the interested reader can read at this web link https://www.cntrobotics.com/robotis-evolving-dynamixel-concept). At present, users have two options for using the SMART DEVICE commands:

1) **Using the TASK V.1 tool** which is part of the older RoboPlus V.1 Suite (http://www.robotis.us/roboplus1/). Fig. 1.6 illustrates the "SMART APP" menu in TASK V.1.

2) **Using the TASK V.2 tool** requires a more elaborate procedure because, starting with TASK V.2.1.4 (March 2017) and for unknown reasons, ROBOTIS removes access to the "SMART DEVICE" submenu for the CM-150's programming interface, however the actual SMART commands **are** still functional within the CM-150 firmware (*at least for now, i.e. April 2019*). Thus to use SMART commands with a CM-150, the user first has to temporarily change the "Programming Controller" field to one of the other Firmware 2.0 controllers (i.e. CM-50/200 or OpenCM-7.00/9.04, see Fig. 1.7 where CM-200 is used).

1.2.2 Organization of Display/Touch Zones on Mobile Device

As far as TASK is concerned, the Mobile Device's Display Screen is divided into 25 zones arranged in a 5x5 grid in either Portrait or Landscape mode (see Fig. 1.9).

When the Mobile Display is used as an Input Device, for example as a Touch Sensor (see more details in Section A3), the R+m.PLAY700 App reports a Zone Number between 1 and 25 (as shown in Fig. 1.9) to the TASK program at run time.

When the Mobile Display is used as an Output Device, for example for a Text item (Line 9 of Fig. 1.8 and see Fig. 1.10 for other functions such as "Shape" and "Number"), these Display Zones can be specified in two ways, depending on the programmer's needs, via a SMART CONSTANT Parameter which can have a fixed value or a variable value during run time:

1.3 Touch Areas and Shapes Display

Let's next look at a basic usage of the Touch Area feature provided with the PLAY700 App which can only monitor **up to 2** Touch Areas simultaneously. We'll also use the Shape Display feature whereas Shape #1 is a Circle, Shape #2 is a Square and Shape #3 is a Triangle (currently, PLAY700 does not allow any user-registered shape to be included in this list).

The program "AD-TouchMe.tskx" essentially monitors the 25 possible Touch Areas (see Fig. 1.9) where the user may have touched the mobile display with one or two fingers, and then it would draw either a circle or a square in those detected zones using randomized colors. Fig. 1.14 describes the algorithm used via an Endless Loop:

1.4 Text-to-Speech and Music Play

This musical project illustrates how to access the Musical Instruments available on the mobile device and how to activate the Text-to-Speech feature:

1) There are 128 Musical Instruments available via the PLAY700 App (from "Acoustic Grand Piano" [1] to "Gunshot" [128]). For each instrument, there are 10 Octave settings [1-10] and 12 Musical Scales or Notes [1-12]: e.g. Note 1 is "Do", while Note 3 is "Re", and so forth until Note 12 which is "Shi". A 3-byte SMART CONSTANT is associated with the Music Function where the lowest byte (Byte 1) contains the Note's value, while the next higher Byte (Byte 2) contains the Octave's value and Byte 3 contains the Instrument's value. The 4th byte of a typical SMART CONSTANT is not used for the Music Function (see Section 1.2.2) and needs to be set to zero.

2) The Text-to-Speech feature uses the same list of Text Items used for displaying them on the mobile screen (see Section 1.2.2) but activates a different SMART DEVICE function.

```
20  ENDLESS LOOP
21  {
22     //  Using Musical Instrument on Mobile Device for 3 Noted Do-Re-Mi - going through 6 different types of piano
23     LOOP FOR ( I = 1 ~ 6 )
24     {
25        SMART: Text Display = [Position:(3,2)],[Item:7],[Size:40],[Color:Green]
26        SMART: Text Display = [Position:(3,3)],[Item:I],[Size:40],[Color:Yellow]
27        SMART: Text to Speech (TTS) = Text Item 7
28        WAIT WHILE ( SMART: Text to Speech (TTS) != 0 )
29        SMART: Text to Speech (TTS) = I
30        WAIT WHILE ( SMART: Text to Speech (TTS) != 0 )
```

Fig. 1.16 Part 1 of Main Algorithm for program "AD-MusicPlay.tskx".

Fig. 1.17 shows Part 2 of the main Endless Loop for the program "AD-MusicPlay.tskx":

1) Line 32 save the current value of Index "I" to Parameter "InstrumentType" and Line 33 shifts this value 2 bytes to the left by multiplying it with the binary number "1 0000 0000 0000 0000" (i.e. to Byte 3 position, see Section 1.2.2 to review details about the structure of the 4-byte SMART CONSTANT). The decimal equivalent of the binary number "1 0000 0000 0000 0000" is "65536" (which was used in

Section 1.2.2). This shifted "InstrumentType" value is then saved in Parameter "InstrumentValueTemp1".

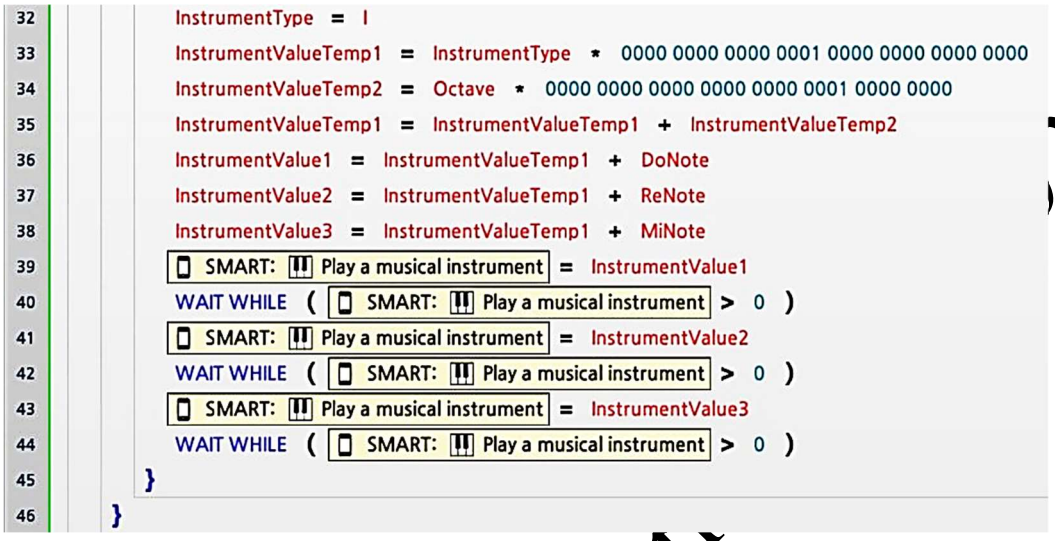

Fig. 1.17 Part 2 of Main Algorithm for program "AD-MusicPlay.tskx".

Video 1.5 shows how the "Music" PLAY700 Project is set up on the mobile device and the run time performance of "AD-MusicPlay.tskx".

1.5 Remote Control using Touch Areas and Sound Effects

In this project, we'll use the Dump Truck (Robot 15) from the DREAM II School Set (see Fig. 1.18).

For this RC project, we'll again use the Touch Area feature but with two different user-interaction schemes:

1) For the standard Direction inputs such as Forward-Backward-Left-Right, the user needs to keep pressing on the selected Touch Area(s) for the robot to keep on performing the user-wanted maneuvers (i.e. "function activation upon pressing on Touch Area").

2) The user can press on two other Touch Areas to Increase or Decrease the robot's Speed independently of the current Direction input (subject of course to a maximum of two Touch Areas at run time), however the Speed change will occur only after the user lifts the finger off the selected Touch Area (i.e. "function activation upon release of Touch Area"). In other words, the user would need to "tap" a given Touch Area to make its related function to work.

We'll also implement two types of Sound Effects:

1) Using the CM-150 Buzzer to confirm a Speed Change.

2) Using the mobile device's Audio Playback service to play an audio clip of an engine running whenever the robot is in motion.

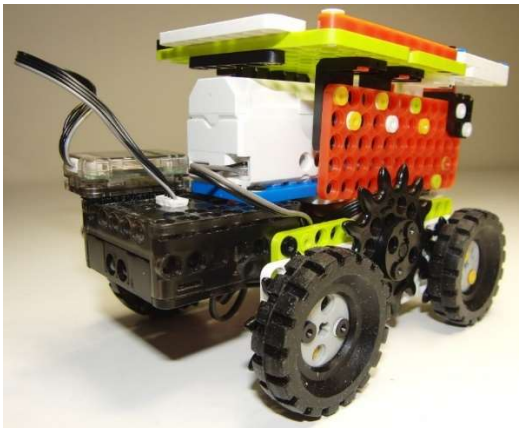

Fig. 1.18 Dump Truck robot from DREAM II School Set.

```
5   PORT[3]:Servo Motor Drive Mode = TRUE (1)
6   PORT[3]:Servo Motor Speed = CCW:1000 (97.75%)
7   PORT[3]:Servo Motor Position = 512
8   Speed = 900
9   TimeDelay1 = 100
10  TimeDelay2 = 25
11  // Getting remote device ready using SMART commands
12  SMART: Screen Rotation = Portrait Mode (1)
13  SMART: Text Display = 0
14  // User needs to make sure that robot is connected to mobile device. User whistles loudy when ready.
15  WAIT WHILE ( SMART: Noise (dB) < 50 )
16
17  // Display Control Text FORWARD, BACKWARD, LEFT, RIGHT
18  SMART: Text Display = [Position:(3,2)],[Item:1],[Size:100],[Color:White]
19  SMART: Text Display = [Position:(3,4)],[Item:2],[Size:100],[Color:White]
20  SMART: Text Display = [Position:(2,3)],[Item:3],[Size:100],[Color:White]
21  SMART: Text Display = [Position:(4,3)],[Item:4],[Size:100],[Color:White]
22  // Display Speed Control Text SLOW or FAST
23  SMART: Text Display = [Position:(1,1)],[Item:5],[Size:75],[Color:Red]
24  SMART: Text Display = [Position:(5,1)],[Item:6],[Size:75],[Color:Red]
25  // Display Platform Control Text RAISE or LOWER
26  SMART: Text Display = [Position:(1,5)],[Item:7],[Size:75],[Color:Yellow]
27  SMART: Text Display = [Position:(5,5)],[Item:8],[Size:75],[Color:Yellow]
```

Fig. 1.19 Initialization Event for "AD-TouchRC-DumpTruck.tskx".

Fig. 1.20 describes Part 1 of the main algorithm implemented as an Endless Loop:

1) The values for the Detected Zones [1-25] for "Touch Area 1" and "Touch Area 2" are respectively saved in Parameters "Touch1" and "Touch2" (Lines 31-32).

2) When either Parameter Touch1 or Touch2 has a non-zero value (Line 33), this means that the user has pressed on the mobile screen somewhere, the algorithm goes on to check whether any Touch Area corresponding to the "Directional" zones has been confirmed using 4 PARALLEL IFs (Lines 35, 41, 47 and

53). In Thai (2018), the PARALLEL IFs structure is explained in detail from design goal to run-time performance, thus that information won't be repeated here. The gist of this structure is to allow the user to combine two "directional" inputs at the same time: for example, "Forward" and "Right" would make the robot make a wider right turn than if only "Right" was used.

3) Lines 37, 43, 49 and 55 are used to add a Sound Effect of a running engine out on the Play Audio 1 channel of the mobile device any time that the robot is set in motion. Please note that the PLAY700 App has a second Audio channel that can be used simultaneously with the first.

4) Finally, please note that the "directional" control of the robot is implemented with the "function activation upon pressing on Touch Area" scheme.

1.6 Voice Control

For this project, we continue to use the Dump Truck as the demonstration platform (see Fig. 1.18) and allows for the same robot functionalities as described in the previous Section 1.5 (see Fig. 1.19), but this time these robot functions are activated by Voice Commands via the Speech Recognition tool of the PLAY700 App. Furthermore, because of the lengthy process of Speech Recognition (using the Web-based Google Speech Engine), each Voice Command will only trigger the corresponding robot function for a short time period, then the robot is reset to its full-stop state waiting for the next Voice Command.

Fig. 1.24 describes the Initialization segment of the program "AD-VoiceControl-WithDelays.tskx":

1) Lines 6-8 initialize a Servo Motor on Port 3, and Lines 9-14 initialize Parameters Speed, TimeDelay1, TimeDelay2, logical flag InvalidCommand and text item VoiceCommand.

2) Next, the mobile device screen is set to Portrait Mode (Line 16), clears the screen (Line 17) and waits for the user to make some noise louder than 50 dB (Line 19) which then triggers Line 20 setting the robot to its full-stop state.

3) Line 21 calls the Menu Function which is described in Fig. 1.25 which lists the TASK code on the left and the resulting mobile user interface on the right. The "Directional" Text Items are set up with Lines 145-149, while the "Speed Control" Text Items are set up by Lines 151-152 and the "Platform Control" Text Items are set up by Lines 154-155. A new Voice Control Text Item for "Talk" is also displayed at Position [3,5] (Line 157). A PLAY700 project is also needed to be set on the mobile device with the following eleven Text Items (in order from 1 to 11): "Forward", "Backward", "Left", "Right", "Slow", "Fast", "Raise", "Lower", "Talk", "Stop" and "Invalid Command".

```
1   //  Voice Control of Dump Truck Maneuvers (CM-150)
2   //  WITH time delays and stop for each command
3   //  By C. N. Thai  3/20/2018
4   START PROGRAM
5   {
6      PORT[3]:Servo Motor Drive Mode  =  TRUE (1)
7      PORT[3]:Servo Motor Speed  =  CCW:1000 (97.75%)
8      PORT[3]:Servo Motor Position  =  512
9      Speed  =  900
10     TimeDelay1  =  500
11     TimeDelay2  =  250
12     TimeDelay3  =  125
13     InvalidCommand  =  FALSE (0)
14     VoiceCommand  =  Text Item 10
15     //  Getting remote device ready using SMART commands
16     SMART:  Screen Rotation  =  Portrait Mode (1)
17     SMART:  Text Display  =  0
18     //  User needs to make sure that robot is connected to mobile device. User whistles loudy when ready.
19     WAIT WHILE  (  SMART:  Noise (dB)  <  50  )
20     CALL  Stop
21     CALL  Menu
```

Fig. 1.24 Initialization Event for "AD-VoiceControl-WithDelays.tskx".

```
140 FUNCTION  Menu
141 {
142    SMART:  Screen Rotation  =  Portrait Mode (1)
143    SMART:  Text Display  =  0
144    //  Display Control Text FORWARD, BACKWARD, LEFT, RIGHT, STOP
145    SMART:  Text Display  =  [Position:(3,2)],[Item:1],[Size:60],[Color:White]
146    SMART:  Text Display  =  [Position:(3,4)],[Item:2],[Size:60],[Color:White]
147    SMART:  Text Display  =  [Position:(2,3)],[Item:3],[Size:60],[Color:White]
148    SMART:  Text Display  =  [Position:(4,3)],[Item:4],[Size:60],[Color:White]
149    SMART:  Text Display  =  [Position:(3,3)],[Item:10],[Size:60],[Color:Red]
150    //  Display Speed Control Text SLOW or FAST
151    SMART:  Text Display  =  [Position:(1,1)],[Item:5],[Size:60],[Color:Red]
152    SMART:  Text Display  =  [Position:(5,1)],[Item:6],[Size:60],[Color:Red]
153    //  Display Platform Control Text RAISE or LOWER
154    SMART:  Text Display  =  [Position:(1,5)],[Item:7],[Size:60],[Color:Yellow]
155    SMART:  Text Display  =  [Position:(5,5)],[Item:8],[Size:60],[Color:Yellow]
156    //  Display Voice Control Text TALK
157    SMART:  Text Display  =  [Position:(3,5)],[Item:9],[Size:60],[Color:Blue]
158 }
```

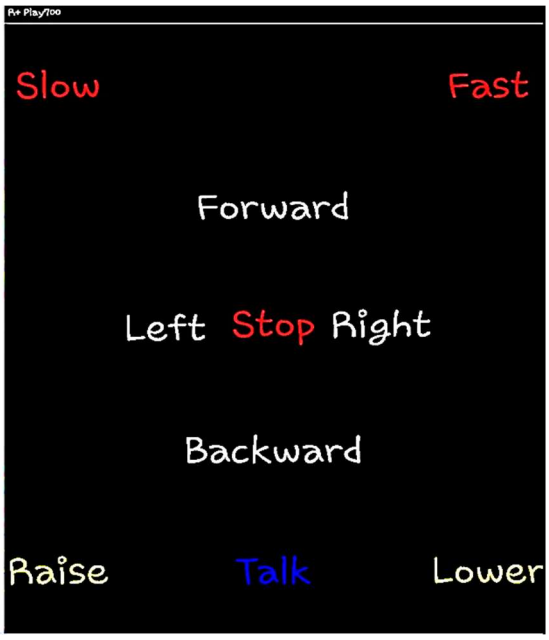

Fig. 1.25 Function "Menu" for "AD-VoiceControl-WithDelays.tskx".

1.7 Bull Fight using NIR and Color Sensors with Audio Files

This Bull Fight project is a remake of a similar project described in Section 6.6 of Thai (2018) which only used the musical melodies from the built-in buzzer for sound effects. But this time around, we'll be using more realistic MP3 sound clips of a growling bull and of an "Ole" exclamation from the crowd attending a typical bullfight. These two sound clips come from the Sound Bible web site (http://soundbible.com/suggest.php?q=bull&x=0&y=0), courtesy of Mike Koenig and as a Creative Commons Attribution 3.0 credit.

Hardware wise, this Bull Robot (Fig. 1.31) illustrates a dual-range object detection scheme whereas:

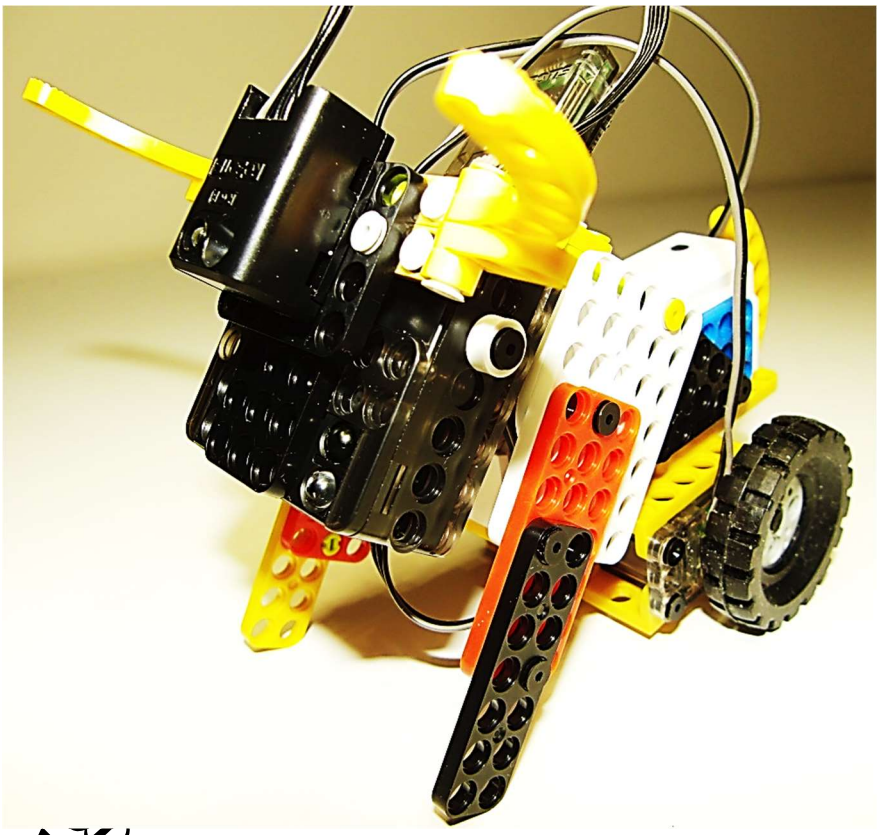

Fig. 1.31 Bull Robot with Color Sensor facing forward.

• For the farther range, the built-in Left and Right NIR Sensors are used to detect the presence of the object and to make the robot move closer to the object. During these maneuvers, the PLAY700 App plays a growling-bull MP3 file out of Audio Channel 1 of the mobile device.

• When the Color Sensor on Port 4 gets in range, then it is used to determine whether the object has a Red color or not. If a Red color is found, the Bull robot would back up and then charge ahead while growling out of Audio Channel 1. At the end of this charge, the PLAY700 App plays the "Ole" MP3 file out of Audio Channel 2 of the mobile device. For any other color found, the robot just stops and plays Melody 20.

1.8 Camera Functions

Using the R+m.PLAY700 App, both rear and front cameras for a typical mobile device are accessible via SMART DEVICE commands inside a TASK program (please check program "AD-SelectCamera.tskx") along with picture and video capturing functions (see Fig. 1.36). Furthermore, some rudimentary image processing tools are also available through TASK programming:

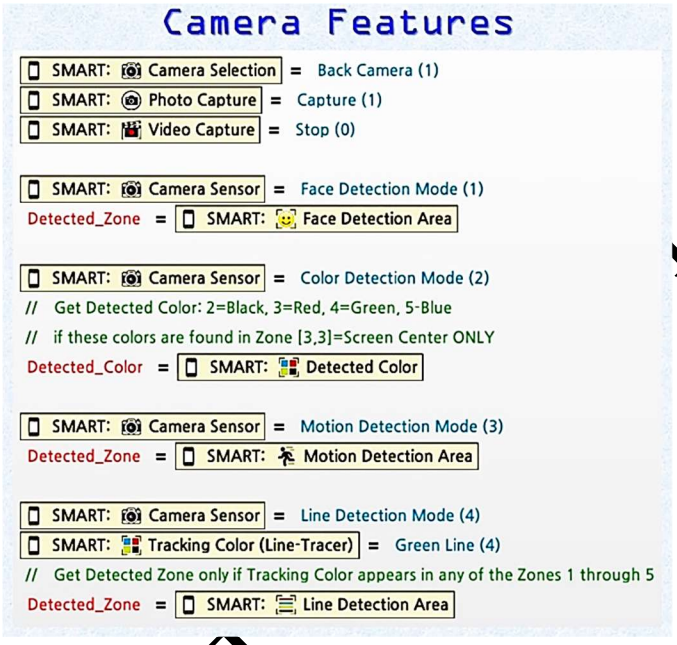

Fig. 1.36 R+m.PLAY700 Vision-related Functions available via TASK Programming.

1) In a "Face Detection" mode, the SMART "Face Detection Area" Function would return a number between 1 and 25 to the TASK program, representing the Zone Number where the "face" was detected (please review Fig. 1.9 for a map of these zones and try out the program "AD-FaceDetection.tskx")

2) In a "Color Detection" mode, only pixels within Zone 13 (i.e. Screen Center or Position [3,3]) are evaluated for their color values to see if they fit, on the average, "fixed" internal numerical RGB criteria for 4 possible color values: 2 for Black, 3 for Red, 4 for Green and 5 for Blue. Please try out program "AD-ColorSensing.tskx" and note that the App would classify a "Yellow" object as being "Red", so this App is not quite robust as users may expect.

3) In a "Motion Detection" mode, the SMART "Motion Detection Area" Function would return a number between 1 and 25 to the TASK program representing the Zone Number where "Motion" was detected (please try out the program "AD-MotionDetection.tskx").

4) In a "Line Detection" mode, the user needs to specify which color the App should be tracking for: 2 for Black, 3 for Red, 4 for Green and 5 for Blue. Furthermore, only the top 5 Screen Zones, i.e. Zone 1 through 5, are considered by the App, thus the SMART "Motion Detection Area" Function would return a number between 1 and 5 to the TASK program if the user-chosen color was found among those zones (please try out the program "AD-ColorLineDetection.tskx").

1.8.1 Picture and Video Capture

This project "AD-CameraCapture.tskx" integrates the following SMART features into a more utilitarian application: text display, touch area, camera selection, picture/video capture, and activating an external App (i.e. Gallery) to review the recorded pictures and videos. The corresponding PLAY700 App project uses the following list of Text Items:

1 – Front Camera

2 – Back Camera

3 – Pictures

4 – Videos

5 – Selected

6 – Taking

7 – First Picture

8 – Second Picture

9 – Gallery

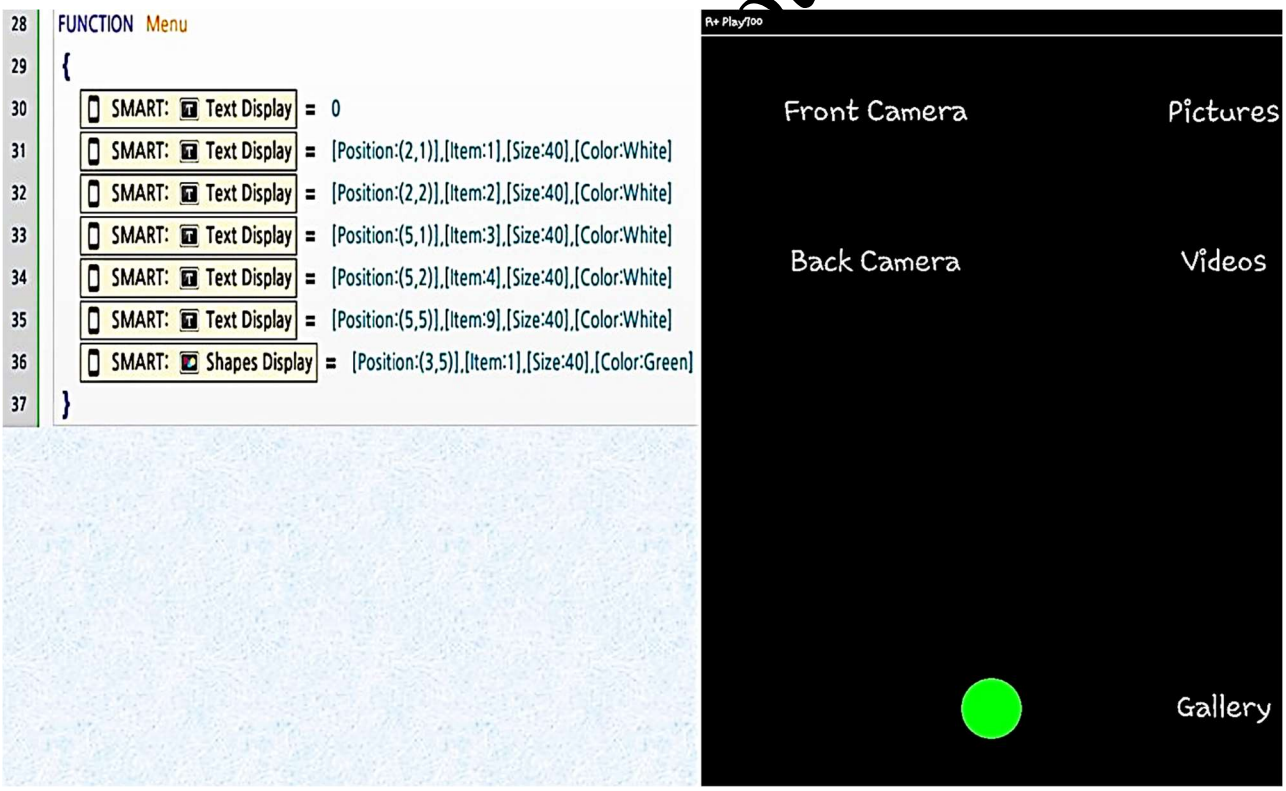

Fig. 1.38 Source Code for Function "Menu" and how it looks at runtime.

1.8.2 Color Dowel Alert

This project "AD-ColorDowelAlert.tskx" showcases a rudimentary Image Processing tool using the following SMART features: text and shape displays, touch area, switching between the three Camera Sensor modes of Motion Detection, Line Detection and Color Sensing with the goal of highlighting the mobile screen's zone where a user-chosen colored dowel can be found in real-time (*as much as possible*). The corresponding PLAY700 App project uses the following list of Text Items:

1 – Black

2 – Red

3 – Green

4 – Blue

5 – Scan

6 – Found

7 – Dowel of Interest

8 – Motion Detected at Red Circle

Fig. 1.46 describes the Initialization section of program "AD-ColorDowelAlert.tskx":

1) Lines 5 to 8 initialize three Parameters "ColorType" (Black, Red, Green or Blue), ScanMode (True/False) and ScanDone (True/False).

2) The robot then waits for the user to make a noise louder than 50 (Line 10).

3) Next, the mobile device is set to use the Back Camera, to set the screen in Portrait Mode, and clears the Text Display overlay (Lines 12-14).

4) Function "Menu" is called at Line 15. Details of Function "Menu" are shown in Fig. 1.48.

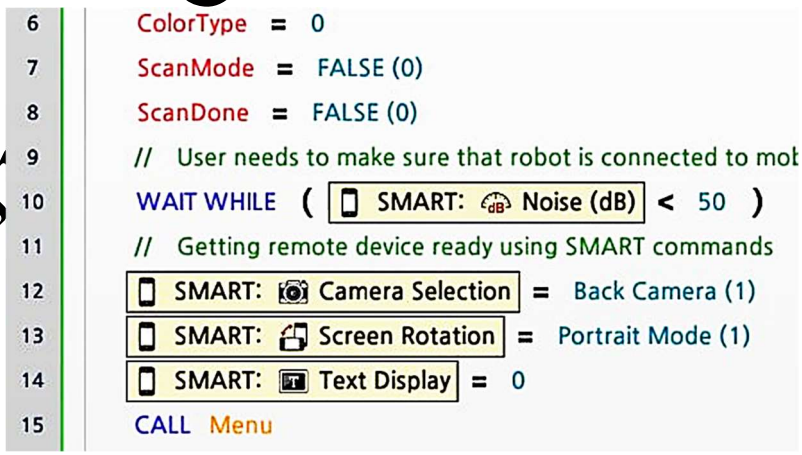

Fig. 1.46 Initialization Part of Main Program in "AD-ColorDowelAlert.tskx".

1.9 Dual CM-150 Programming via Bluetooth

Eventually, the reader would find the two GPIO Ports (3 and 4) on the CM-150 as quite limiting and one may wonder if a robot with multiple CM-150 controllers can be built and most importantly programmed to work together collaboratively. And the answer is Yes, but for now between 2 CM-150s only, when using Bluetooth (BT-210 or BT-410 – in theory) because so far ROBOTIS has only released Bluetooth firmware that can only deal with a 1 to 1 pairing scheme, Master-Slave to be exact, and with a 1 to N version to be released sometime in the future (http://emanual.robotis.com/docs/en/parts/communication/bt-210/, http://emanual.robotis.com/docs/en/parts/communication/bt-410/).

The author would add that, presently when using TASK, ones can coordinate **multiple** ROBOTIS CM-XXX controllers to work together ONLY using ZigBee communications hardware (http://emanual.robotis.com/docs/en/parts/communication/zig-110/, http://www.robotis.us/zig-110a-set/) – unfortunately these ZigBee hardware are discontinued so they are hard to obtain nowadays. Thai (2017) devoted Chapter 7 of that book to the topic of "Communication Programming with REMOCON Packets" and the concept of "Message-Shaping" as it applied to ZigBee communications hardware and protocols (i.e. 1 to 1 and broadcast modes). This section 1.9 applies this "Message-Shaping" concept to a Bluetooth Master-Slave configuration between two CM-150 controllers.

Currently, ROBOTIS offers a set of Master/Slave BT-210s (http://www.robotis.us/bt-210-set/) and another set of Master/Slave BT-410s (http://www.robotis.us/bt-410-set/), however the author has found that only the **BT-210 Master/Slave set** would work properly for TASK codes between 2 CM-150s as developed in this Section 1.9. If the reader happens to have 2 BT-210s available and both are in Slave mode, then the user can follow the procedure described at this web link (http://emanual.robotis.com/docs/en/parts/communication/bt-210/) to convert them into a Master/Slave set. Unfortunately, the reader will also need to use an OpenCM-9.04-C controller (http://www.robotis.us/opencm9-04-c-with-onboard-xl-type-connectors/) in order to perform the necessary steps to set one BT-210 into a Master mode and also to pair it with the other BT-210 in Slave mode.

1.10 Use of Sample Codes on Mobile Devices

All sample TASK codes for this book can also be transferred to an Android or iOS device so that they can be used by the mobile-equivalent of the TASK tool, called R+m.TASK2, available at (https://play.google.com/store/apps/details?id=com.robotis.task2&hl=en) for the Android version or at (https://itunes.apple.com/us/app/r+m-task2-robotis/id1031166481?mt=8) for the iOS version. To transfer the sample TSK codes to an Android device, the user only needs to use Windows Explorer on the PC side and then drags and drops them into the appropriate folder on the mobile device (see Fig. 1.58). For an iOS device, the user will need to use iTunes to transfer the sample codes from the PC to the user personal iOS device, via File Sharing (see Fig. 1.59).

Chapter 2: Using Edbot® Dream Pro

Circa 2015, the Edbot software tool was first created as a server/client tool to be used within an actual classroom environment, along with teacher supervision of student access to individual ROBOTIS MINI robots (http://www.robotis.us/robotis-mini/), programmed with the MIT Scratch 2 Offline IDE (http://ed.bot/edbot/). In 2018, the Edbot suite was expanded with the release of the Edbot Dream tool (http://ed.bot/edbotdream/) which was designed to work with the DREAM II School Set (http://www.robotis.us/dream/).

In January 2019, SCRATCH 3 (https://en.scratch-wiki.info/wiki/Scratch_3.0) was released with an online interface and a desktop interface (https://scratch.mit.edu/download). In March 2019, Edbot V.5 was released allowing the additional usage of SCRATCH 3 via a web interface (http://scratch.ed.bot/) – a desktop version may be released in the future. Unfortunately, this web interface yields a slower communications speed with the DREAM robot and makes autonomous behaviors harder to achieve, thus this book concentrates on the usage of SCRATCH 2 Offline Editor instead. Please see Section 2.5 for more discussions regarding the use of SCRATCH 3 with the DREAM II robot.

Furthermore, the Edbot Dream Pro tool (V.5 and above) allows **multiple robot** programming via three software interfaces: Scratch, Python and JavaScript (http://support.ed.bot/edbot-dream-index.html), and with more languages such as Node.js, Java and C# and other platforms such as Chromebooks being planned.

In this book, an individual user is assumed, i.e. the user's PC is used as the local host (Port 8080) and the Edbot Dream Pro tool is used in its server mode. Please view this YouTube video for an overview of the capabilities of the Edbot Dream Pro tool (https://www.youtube.com/watch?v=pOk5szsngM0).

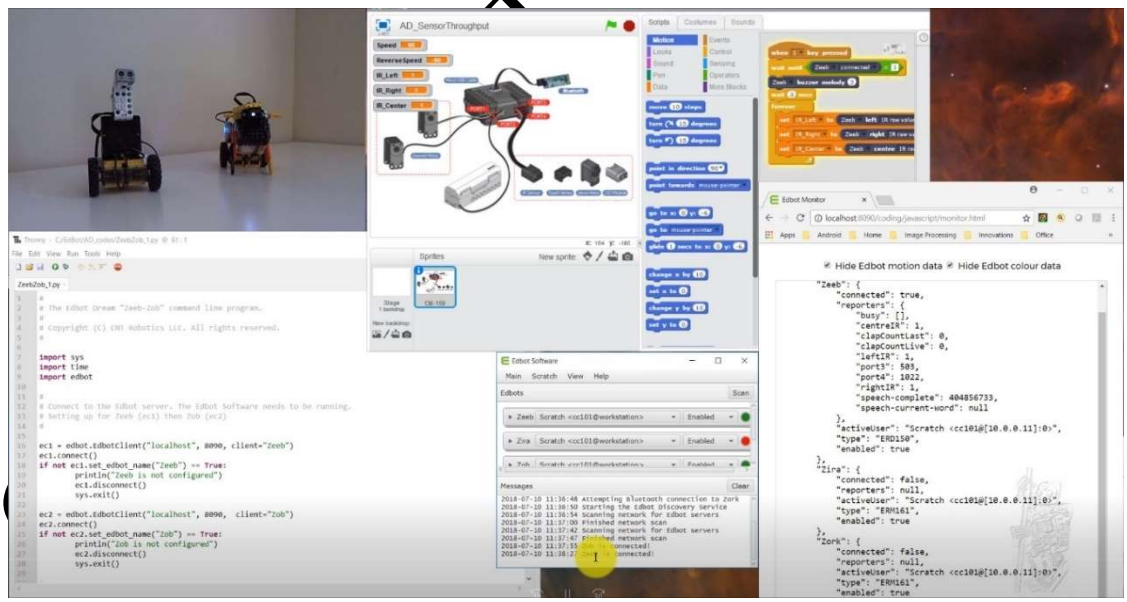

Fig. 2.1 Multi-software and multi-robot application of the Edbot Dream Pro tool.

Furthermore, the Edbot software environment allows the mixing of MINI and DREAM robots in a single instance of Scratch (https://www.youtube.com/watch?v=yUpx42eNR2I).

For this Chapter, the author assumes that the reader is already familiar with MIT SCRATCH 2 software or has read Chapter 4 of the other DREAM book by the author (Thai, 2018). If not, the user is recommended to first read up on such works as Ford (2014), Warner (2015) or Vlieg (2016) to have a thorough understanding of the SCRATCH 2 language. In this Chapter, only selected SCRATCH 2 features, most applicable to robotics and unique to the workings of the Edbot Dream Pro software, would be presented. However, Section 2.5 will present some "appropriate" SCRATCH 3 applications with the DREAM II robot.

2.1 Edbot's Hardware and Software Installation Requirements

The user will need to purchase the appropriate Edbot Dream kits and/or software product keys from Robots in School Ltd. (https://shop.ed.bot/collections/products). If the user has already purchased the DREAM II School Set elsewhere, the user can just buy the Edbot Pro product key from Robots in School Ltd. (https://shop.ed.bot/collections/products/products/edbot-software) or from ROBOTIS USA (http://www.robotis.us/edbot-software-product-key/). **In this book, the author assumes that the user is using the Pro version of the Edbot Dream tool (Version 5.0.6.1280 or higher).**

Hardware wise, the Edbot Dream software only works with either the ROBOTIS BT-210 module (http://www.robotis.us/bt-210/), or the BlueTooth module designed by Robots in School Ltd. (https://shop.ed.bot/collections/products/products/bluetooth-module-for-robotis-dream).

Please view Video 2.0 for the typical steps needed to install the Edbot Dream Pro software on the user's PC. **Importantly, each software product key will be bound permanently to the specific BT-210 or the Edbot BlueTooth modules used, thus the user will need to plan out carefully his or her "robots" organization if the user wants to use the Edbot software with multiple robots.** Fig. 2.2 shows the author's Edbot set up for 4 robots: Zeeb and Zob are DREAM robots, while Zira and Zork are MINI robots.

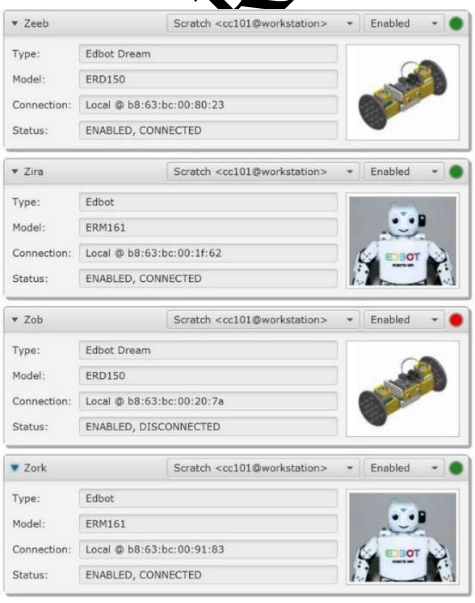

Fig. 2.2 Author's Edbot Setup for his 4 robots.

Lastly, the user needs to update the firmware of the DREAM controller CM-150 to Version 33 or above. Please refer to this YouTube video as a general procedure (https://www.youtube.com/watch?v=LX3WuWJQosw) using the MANAGER tool from ROBOTIS.

2.3 SCRATCH 2 Blocks Provided with Edbot Software

Edbot provides 13 Stack Blocks and 7 Reporter Blocks which can be accessed under the "More Blocks" item of the SCRIPTS tab (see Fig. 2.8).

2.3.1 MOTOR & SERVO Blocks

A quick reminder to the readers regarding the hardware design of the DREAM Controller CM-150 is that Ports 1 and 2 are reserved strictly for Continuous-Turn motors GM-10A (http://www.robotis.us/geared-motor-gm-10a/), while Ports 3 and 4 can be used with Servo motors SM-10 (http://www.robotis.us/servo-motor-sm-10/) which can also function as Continuous-Turn motors. Additionally, Ports 3 and 4 can be used for various Sensors/Actuators such as NIR sensors IRSS-10 or LED modules LM-10. The reader can browse through a list of these sensors at http://www.robotis.us/sensors/.

Edbot offers two types of motor/servo control:

1. Single-Actuator Blocks – i.e. one block is used to control each individual actuator (see Fig. 2.9). There are 3 implementations of these Single-Actuator (SA) blocks: "Set Motor Speed", "Set Servo On/Off" and "Set Servo Position Speed".

Fig. 2.9 Edbot's SINGLE ACTUATOR (SA) Blocks.

2. Multiple-Actuators Blocks – i.e. one block can be used to control multiple actuators simultaneously (i.e. with ONE "big" communication packet via the Edbot software tool) – see Fig. 2.10. **Please note that this feature only works with Edbot V. 4.1.979 or higher, furthermore the CM-150 firmware needs to be V.32 or higher.**

Fig. 2.10 Edbot's MULTIPLE-ACTUATORS Blocks.

2.3.2 LED & BUZZER Blocks

Fig. 2.11 shows the Edbot blocks used to control the external LED module LM-10 (http://www.robotis.us/led-module-lm-10/) which can be only connected to Ports 3 or 4, and the built-in BUZZER located inside the CM-150 Controller.

Fig. 2.11 Edbot's LED and BUZZER Blocks.

The "Set LED Module" block requires two inputs: a Port Number (only 3 or 4 are allowed) and a numerical value [0-3] to turn on/off the Orange and Blue LEDs of the LM-10 (details can be found at this link http://support.ed.bot/edbot-dream-scratch2.html).

The "Buzzer Pitch Duration" block also requires two inputs: the Pitch (i.e. Musical Scale) [0-48], and its time duration [0-50], each time unit corresponding to 1/10 of a second (thus a maximum of 5 seconds duration).

The "Buzzer Melody" block can take on a numerical input between 0 and 24, and the chosen Melody will be played for 5 seconds by the CM-150 Controller through its embedded speaker/microphone.

2.3.3 SENSOR Blocks

The first SENSOR block is the "Port Sensor" block which is designed to report on the current Calibrated Values of a variety of Sensors that can be attached to either Ports 3 or 4 (see Fig. 2.12). The definition and range of the numerical values reported by this block vary depending on the sensor type and more details are available at this web page (http://support.ed.bot/edbot-dream-scratch2.html):

Fig. 2.12 Edbot's PORT SENSOR Blocks.

2.3.4 SPEECH Blocks

These blocks provide Text-to-Speech capabilities via the Edbot software tool (not via SCRATCH 2). The user can choose the voice of "David" or "Zira" under "Server/Setup" off the main menu bar of the Edbot tool (see Fig. 2.15). The discerning reader may have noticed the check marks for items "Bypass active user" and "Available on network" which will be discussed further in Section 2.6 about Python programming with the Edbot tool.

There are 3 speech-related blocks: "Say", "Say Until Done" and "Current Word" (see Fig. 2.16). The difference between "Say" and "Say Until Done" is that "Say" won't wait for the spoken words to be finished before executing the next SCRATCH block, while "Say Until Done" would wait for the speech to finish before continuing to the next SCRATCH block. "Current Word" provides the words being spoken or an

"empty text" field if nothing is being spoken, and it can be used to provide visual emphasis during speech operations (see example at this web link http://support.ed.bot/edbot-dream-scratch2.html).

Fig. 2.16 Edbot's SPEECH blocks.

2.3.5 GENERAL Blocks

2.4 Using SCRATCH 2 with Edbot Dream Pro

In this section, a variety of SCRATCH 2 projects will be presented in detail with the goal of helping the reader become proficient at using Edbot blocks to perform selected robotics applications with the ROBOTIS DREAM II system. Where appropriate, the performances between a TASK solution and a SCRATCH-Edbot solution will be compared to enhance the readers appreciation of robotics engineering in general.

Due to a long tool-chain (SCRATCH <-> EDBOT <-> WINDOWS OS <-> BT COMM <-> ROBOT) which can create communication delays, SCRATCH applications usually do not perform very well when sensors data are required at a fast rate for autonomous behaviors such as following a random line track, see an example with a PLAY700 robot (CM-50) at this link (https://www.youtube.com/watch?v=lwpsJu5uRu0). However, the combination of Edbot V.4.1.979 (or newer) and Firmware V.32 (or newer) for the CM-150 provides much performance improvement as demonstrated in the following sub-sections.

2.4.1 Sensor Throughput Performance

2.4.2 Motor Control using Single-Actuator (SA) Blocks

2.4.3 Motor Control using Multiple-Actuators (MA) Blocks

To mitigate the timing delays for activating and deactivating motors attached to Ports 1 and 2, one approach is to send a "big data packet" containing commands to both Ports using a single SCRATCH block and once that big packet is received in the local memory of the CM-150, we'll let the CM-150 deliver the actual commands to Ports 1 and 2 as normal. But now, as the command sending process is from the CM-150's local memory, it will have minimal time delays as with the binary TASK code - as described in Section 2.4.2.

Fig. 2.22 displays the Script for Project "AD_Maneuvers_MA.sb2" where "Multiple-Actuators" (MA) blocks are used to send activation and deactivation commands to Ports 1 and 2 "simultaneously". The same sequence of commands are issued to make the robot first go forward and straight at 50% power for 1 second (Lines 5-6), next stop both motors (i.e. the robot) for 2 seconds (Lines 7-8), then make the robot go backward for 1 second at 50% power (Lines 9-10), and finally stop the robot for good (Line 11). Please note that a positive value for the motor speed produces a counter-clockwise rotation of the motor, and vice-versa for a negative value of the motor speed (see Lines 5 and 9).

Fig. 2.22 SCRATCH Script for Project "AD_Maneuvers_MA.SB2".

2.4.4 Variable Speed Motor Control with MA Blocks

Reviewing Fig. 2.22, the reader may have already noticed that the motor speed values were specified as numerical constants (50 or -50) in Lines 5 and 9, but sometimes a Variable Parameter (with such name as "Speed") may have to be used, thus somehow we'll have to incorporate the Numerical Value represented by the String "Speed" into the creation of those "Big Data Packets" used in Lines 5 and 9. The SCRATCH's JOINT Operator can be used for this purpose as shown in Fig. 2.23 for the next SCRATCH project named "AD_Maneuvers_MAV.SB2". In this project, the reader needs to create a User Block (i.e. User Function) named "Move", using the "Make a Block" tool, available under the option "More Blocks" of the SCRIPTS Tab (see Fig. 2.24). If the reader is not familiar with this procedure, please watch the first part of Video 2.4 for a step-by-step tutorial on how to create and define the blocks used in this User Block named "Move".

2.4.5 Avoider (R+SCRATCH vs. Edbot)

The materials in this Section 2.4.5 can be compared to Section 4.5 of the author's first DREAM book (Thai 2018) where the tool chain SCRATCH 2/R+SCRATCH was used, and it could only support Single-Actuator blocks.

Video 2.5 shows that this SCRATCH/Edbot Project makes the robot responsive enough to avoid obstacles in front of it. However, once the obstacle is removed, the robot was still performing avoiding maneuvers for a few more cycles before going forward again as it should. Reviewing Video 2.5 more carefully, the reader would notice that the message "Forward" was displayed quite promptly in the SCRATCH "Stage" area upon the removal of the frontal obstacle. This means that the Edbot tool can handle the "Sensor Data" flow from the CM-150 to the SCRATCH 2 IDE fast enough for autonomous sensor-intensive behaviors. However, Edbot somehow handles the "Motor Commands" flow from the SCRATCH 2 IDE to the CM-150 more slowly, perhaps a larger buffer was used for the SCRATCH to CM-150 data flow? As the author does not have knowledge of the inner workings of the Edbot tool, he cannot say anything for sure.

Next, Project "AD_Avoider-IF_ELSE_IF_2.SB2" was created with the goal of using the "Current NIR Sensor Data" directly as arguments inside the various conditional statements IF and ELSE-IF (see Lines 8, 12, 16 and 20 of Fig. 2.29) with the hope that it would mitigate the previous data flow problems.

Fig. 2.31 Main Script for Project "AD_Avoider-PAR_Loops.SB2".

Lines 9-12 and Lines 13-16 denote the two Parallel REPEAT_UNTIL loops used. The reader should also note that the Conditional Statements used are also simpler (i.e. quicker to execute at runtime) than the ones used by the IF-ELSE-IF structures as shown in Figs. 2.26 and 2.29. Furthermore, "Current" Sensor Data (i.e. Reporter blocks) will need to be used for Loops to work properly (Lines 9 and 13).

Similarly, the Parallel Loops algorithm requires a Time Delay (Line 7) to let the designated actions play out (Lines 12 and 16). Again, the user can tune Parameter "DelayTime" to obtain the optimal overall response for his or her system. For the author's robot, 0.5 second was also about the best value to use, values less or more than 0.5 second made this robot less responsive when the frontal obstacle was abruptly removed.

The Parallel-LOOPs approach also needs the default action ("GoForward") to be laid out as the last action after the Parallel Loops (Lines 17-18).

Finally, the Parallel-LOOPs approach has the distinction of using the least number of blocks than the other two approaches.

Video 2.8 also shows that there is much improvement in responsiveness to random obstacle movements over the IF-ELSE-IF structure.

2.4.6 Yin-Yang Track Follower

In a way, this Section 2.4.6 reverses the logical perspective of the Section 2.4.5 as the DREAM Robot will now use its NIR sensors to try to follow/track a frontal object.

We are now ready to tackle the Yin-Yang Track Following Project. The Yin-Yang Track gets its name from the feature that the track switches from Black to White and vice-versa for half of its total layout area (see Fig. 2.35). Analyzing this Yin-Yang track further, the author realized that more NIR sensors are needed as the background characteristics had to be known in addition to the usual track characteristics to help steer the robot onto the "right track". Thus, two NIR Sensors IRSS-10 were connected to Ports 3 and 4 and mounted to the front of the robot as shown in Fig 2.36.

Section 4.7 of Thai (2018) discussed the details of how to track an arbitrary black track on a white background and the essential algorithm reduced to two mutually exclusive condition/action pairs and one default action:

1. If the Left IR Sensor sees Black and Right IR Sensor sees White, then the robot needs to turn Left to get back to the black line.
2. Else If the Right IR Sensor sees Black and the Left IR Sensor sees White, then the robot needs to turn Right to get back to the black line.
3. Else the robot needs to go Forward (and the robot needs to start its trek straddling on the black line also).

Fig. 2.35 Yin-Yang Track.

Fig. 2.36 Yin-Yang Track Following Robot.

2.4.7 Zeeb-Zob Dowel Search Game

This section assumes that the user has 2 DREAM robots, namely Zeeb and Zob for the author (see Fig. 2.43). Zeeb is essentially the Avoider with 2 added NIR Sensors IRSS-10 on Ports 3 and 4, and Zob is the Buffalo with 1 added NIR Sensor IRSS-10 on Port 3.

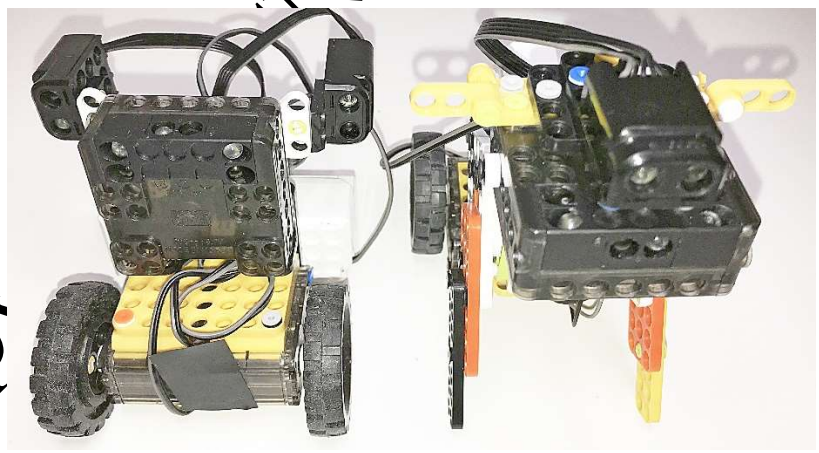

Fig. 2.43 DREAM Robots Zeeb (left) and Zob (right).

The goal of this project is to show how easy it is for Edbot to control two robots concurrently and for SCRATCH to share "control events" between Sprites. The "AD_ZeebZob_DowelSearch.SB2" Project uses

3 Sprites as shown in Fig. 2.44: “Zeeb” and “Zob” represent the respective robots, while the “Cat” represents the Game Controller.

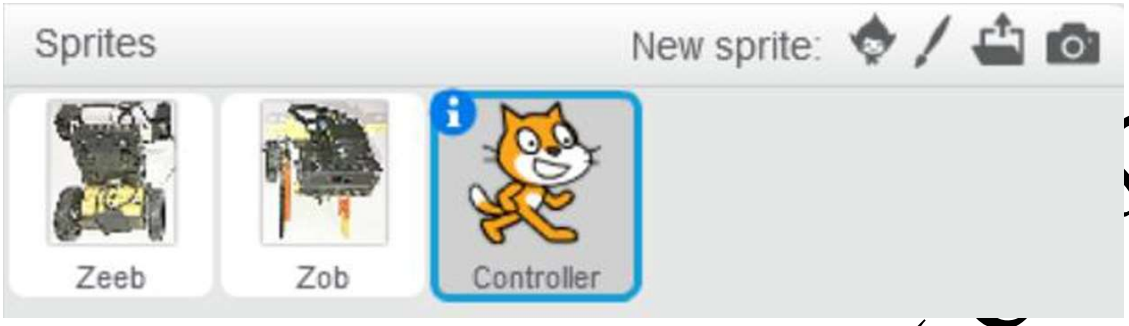

Fig. 2.44 Project “AD_ZeebZob_DowelSearch.SB2” uses 3 Sprites: Zeeb, Zob & Cat Controller.

2.4.9 Using Servo Motor SM-10 in Video Game

This project is an adaptation from a SCRATCH example written by ROBOTIS for the IoT system (http://www.robotis.com/model/page.php?co_id=prd_iot). Currently (Spring 2019), the IoT kit is only available in South Korea. It simulates a Car racing on a Track which is being scrolled up in the Stage Area of the SCRATCH IDE (see Fig. 2.52). The user can steer the Car by using a Servo Motor (SM-10) as a Steering Column (i.e. as a rotational position encoder) (see Fig. 2.53).

Fig. 2.52 Stage Area of SCRATCH IDE for Project “AD-CarGame.SB2”.

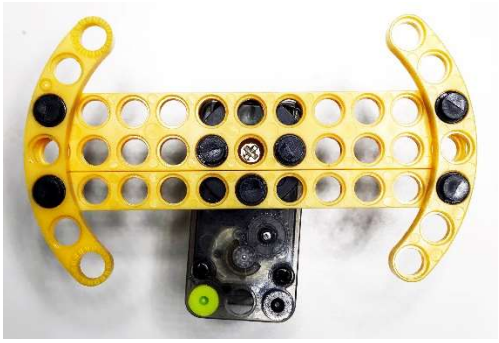

Fig. 2.53 Servo Motor SM-10 used as a "Steering Column" for Project "AD-CarGame.SB2".

2.6 Using PYTHON with Edbot Dream Pro

Using SCRATCH or PYTHON with the Edbot tool provides the same functionalities in controlling the typical Dream robot. However, with Python, the user can get one step closer to the "nuts and bolts" of the Edbot tool and learn some more details of the ROBOTIS firmware used on the Dream Controller CM-150. A PYTHON program will require more steps than a TASK or SCRATCH program to get the robot to do a typical action, but its execution speed is much higher and it will allow the user to tap into the vast resources of other PYTHON packages available at the Python Software Foundation (https://pypi.org/ and https://docs.python.org/3/contents.html). The web site Python Central is also a great resource for learning and practicing Python programming (https://www.pythoncentral.io/).

Edbot V.5 was a major upgrade which changed fundamentally its PYTHON API, thus all sample codes discussed in this section used the Edbot V.5.0.6.1280. The reader is assumed to have some basic skills in Python Programming, if not the author is recommending these books to get the reader better prepared for materials presented in this section (web resources abound for Python also):

- "Python for Tweens and Teens: Learn Computational And Algorithmic Thinking" (Bouras and Ainarozidou, 2017).
- "Python Crash Course", 2nd Edition (Matthes, 2019).

For more in-depth treatments of Python, the user is recommended to refer to these works:

- "Programming in Python 3", 2nd Edition (Summerfield, 2010).
- "The Python 3 Standard Library by Example", 1st Edition (Hellmann, 2017).

Documentation for the Edbot Dream Python API is available at this web link (http://support.ed.bot/edbot-dream-python3.html). This web page also shows how to install Python on the user's PC and provides a short primer on the programming steps needed for controlling the ROBOTIS Crocodile and Scorpion robots. When the Edbot tool was installed, it created a few Python example codes under the user's "Documents\Edbot\python" folder:

- dream_airplane.py
- dream_crocodile.py
- dream_elephant.py
- dream_motorcycle.py
- dream_scorpion.py

2.6.2 Avoider-Follower Applications

In this project "AD_Avoider_Follower_IfElseIf.py", we'll use the Pygame package to monitor the user's keyboard inputs and to provide graphical screen outputs at run time. The web site "https://www.pygame.org/docs/" has all the official PyGame documents that the reader can consult as needed. If the reader prefers a regular textbook, the author would recommend the work by Kinsley & McGugan (2015).

The Avoider robot named Zeeb is used in this project and see Fig. 2.70 for a screen capture at run time.

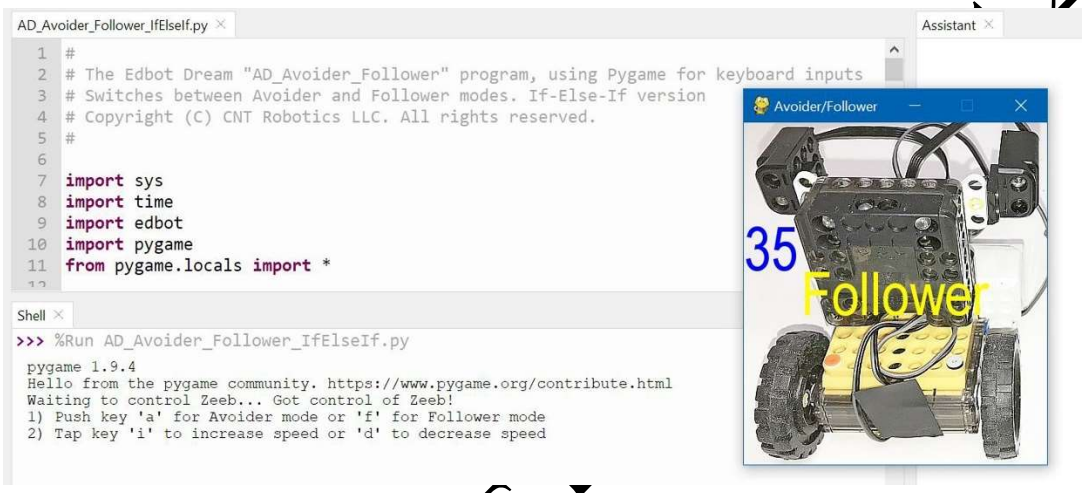

Fig. 2.70 Screen capture at run time for "AD_Avoider_Follower_IfElse_If.py".

2.6.4 Dowel Search Game using Threads & Events

This project had been previously described in Section 2.4.7 for its SCRATCH implementation which was using the concepts of Event Programming via "Message Broadcast" and "Message Received" between 3 independent Sprites (Zeeb, Zob and Cat as the Game Controller). The same Event Programming approach will be used in this Python project using Python-equivalent constructs which are "Thread" and "Event" objects from the "**threading**" module of the Python Standard Library (https://docs.python.org/3/library/threading.html#). The following web page was very helpful to the author by providing working code snippets used in this project: https://pymotw.com/3/threading/index.html#module-threading – it is maintained by Doug Hellmann.

This Python project "AD_ZeebZob_DowelSearch.py" uses the same approaches as previous Python projects regarding Edbot robot control and PyGame keyboard/graphics interfacing so only the newer concepts of threading and event signaling will be described further in the following paragraphs. If the reader is interested in textbooks to learn more about "threads" or "concurrency" in general, the author would suggest Hellmann (2017) and Nguyen (2018).

AD_ZeebZob_DowelSearch.py

```
#
# The Edbot Dream "AD_ZeebZob_DowelSearch" program, using Pygame for keyboard inputs
# Using threads and event objects for signaling scheme
# Copyright (C) 2019 - CNT Robotics LLC. All rights reserved.
#

import sys
import time
import random
import edbot
import pygame
from pygame.locals import *
import threading

#
# Functions definitions start here
#
def forward(robot_name):
    global speed, r_speed
    ec.set_servo_speed(robot_name, "1/" + str(speed) + "/2/" + str(r_speed))
```

Shell

```
Getting out of zob_search()
winner is 2

Getting out of game_end()

Getting out of zeeb_search()
winner is 2
```

Fig. 2.86 Runtime Display for "AD_ZeebZob_DowelSearch.py".

2.6.5 Using USB Camera with OpenCV & NumPy

Another advantage when using Python which is a widely adopted language is the large repository of modules/tools found at the Python Software Foundation https://pypi.org/. This section 2.6.5 would illustrate the use of the OpenCV module and a related module called NumPy (https://pypi.org/project/opencv-python/). If the reader is using Thonny, it is recommended to use the "Manage Package" tool to search for the module "opencv-python" and install it. This process will install all related modules for the reader in one step. The author is using OpenCV-Python V.4.1.0.25 and a Logitech webcam C910 for the example programs in this section (see Fig. 2.90).

Fig. 2.90 Zeeb robot with webcam C910.

The field of Computer Vision or Machine Vision is large and complex, thus in this section the author is only trying to show that, with a bit of work, the reader can incorporate Machine Vision capabilities to a "lowly" DREAM robot. References for Machine Vision, OpenCV and NumPy abound, so the author is only recommending a few that he is familiar with:

- "Learning OpenCV 3" by Kaehler and Bradski (2017), this book is written for the C/C++ programming language but it is a thorough reference to keep.
- For Cookbook types of OpenCV book references in Python, the author has used "Learn OpenCV 4 by Building Projects" by Escrivá and Mendonça (2018) and "OpenCV 3 Computer Vision with Python Cookbook" by Spizhevoy and Rybnikov (2018) and had found them useful.
- Web resources (free and at cost) for OpenCV-Python and NumPy can be found at many places and here are a few:
    - https://opencv-python-tutroals.readthedocs.io/en/latest/index.html
    - https://www.learnopencv.com/about/
    - https://www.pyimagesearch.com/
    - http://www.numpy.org/ and https://docs.scipy.org/doc/numpy/.
    - http://cs231n.github.io/python-numpy-tutorial/

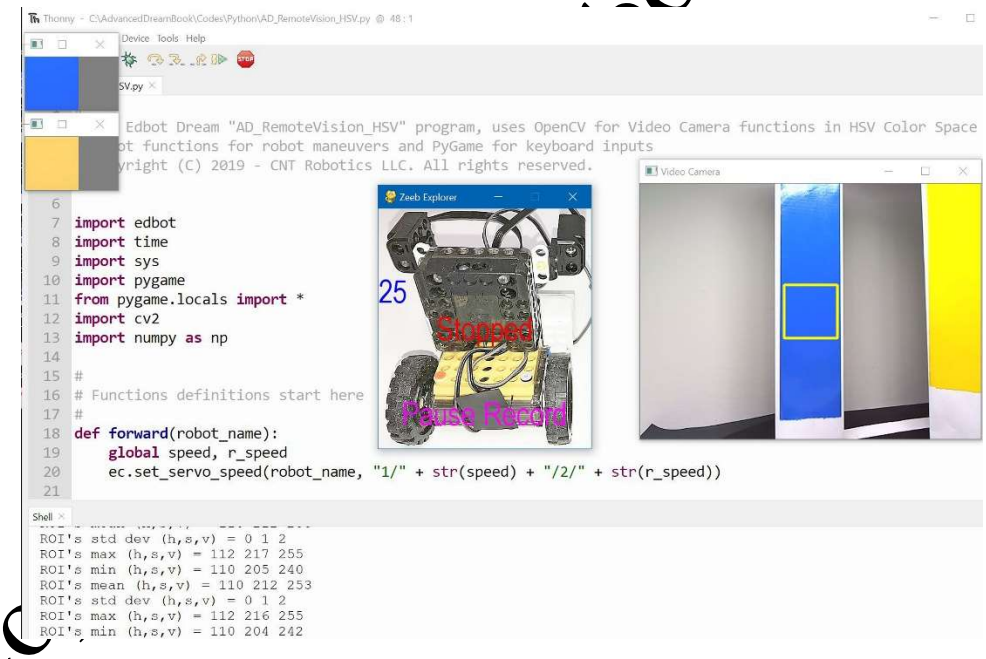

Fig. 2.91 Run-time Displays for "AD_RemoteVision_HSV.py".

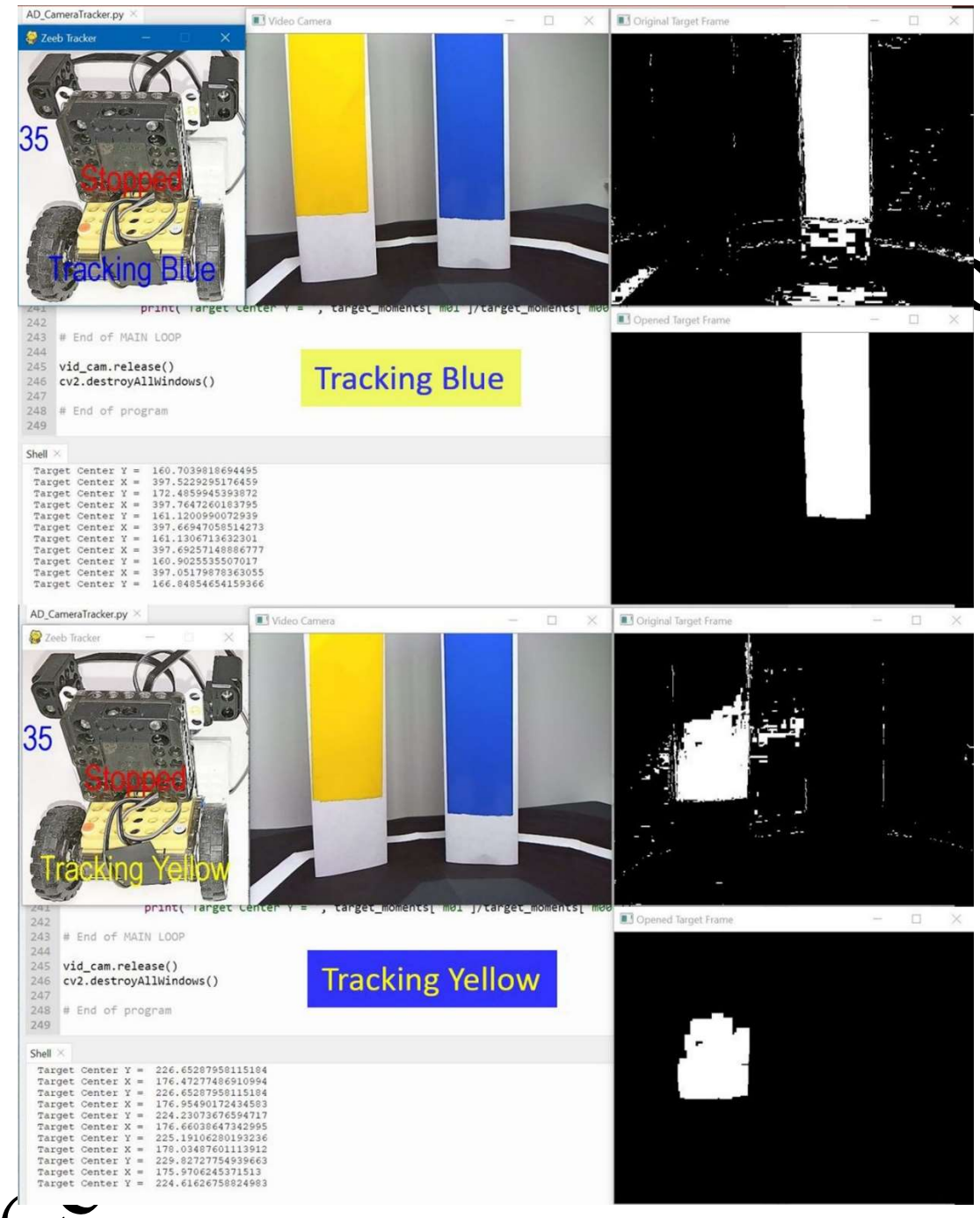

Fig. 2.102 "AD_CameraTracker.py" can be used to track "Blue" or "Yellow" objects.