Chapter 1: Foundational Concepts

In 2024, ROBOTIS released KINDER which is a programmable robotic kit. It is part of the "new" OLLO Series which includes other kits, such as J, K, INITIATE, SPARK, EXCEL, and PRO, which are designed for users at different age groups or robotics knowledge levels.

KINDER is designed for young robotics beginners in elementary school students. It comes with an Integrated Controller named RB-86 (https://emanual.robotis.com/docs/en/parts/controller/rb-86/) and an assortment of plastic frame parts. Although the KINDER's User Manual mentions a Bluetooth Remote Controller named RC-300 in the back of its front cover, *this Physical Remote Controller does not come with the kit itself*. But we can still do Remote Control projects if we use the Virtual RC-300 from the R-BLOCK App via Bluetooth services already available on personal computers and mobile devices.

The RB-86 combines several mechatronics features into a single "brick" (see Fig. 1.1):

- A micro-controller programmable with the R-BLOCK IDE (accessible via a web browser or as a mobile application).
- 2 internal electrical motors to provide a single rotational axle with independent Left-Right motor controls.
- 6 Near-InfraRed (NIR) sensors to detect objects or to track arbitrary black lines.
- 2 Dynamixel connectors to allow the programmer's access to external actuators, such as the XL-330 (not included with the KINDER kit).
- 2 Controllable LEDs.
- An internal Bluetooth LE module that allows the RB-86 to communicate with R-BLOCK and its Virtual RC-300 (via Desktops or Single Board Computers), and also with the physical remote controller RC-300/100B (https://emanual.robotis.com/docs/en/parts/controller/rb-86/#connect-to-remote-controller).

Fig. 1.1 KINDER's Controller RB-86.

This chapter's goal is to **introduce** the reader to Foundational Robotics Concepts that are embodied within the **Hardware and Software Design** approaches used by ROBOTIS in the KINDER kit – but at a high level, just to provide an overview of the **Main Ideas**. Later, these Concepts will be revisited in depth and in breadth within later chapters of the book in a scaffolding flow that is aimed to be useful and efficient for self-learners.

1.1 ROBOTIS Hardware Design Approach

ROBOTIS uses a Rivet system to fasten different types of mechanical parts together to design/build their robots (see Fig. 1.2):

- The sturdier 7 mm DIA rivet system is better suited to motor skill levels of younger children (supplied with the KINDER kit). However, please note that **ROBOTIS materials** refer to this bigger rivet as the **12 mm** rivet (12 mm being the distance between adjacent mounting holes).
- The smaller 3.5 mm DIA rivet system requires more manipulation skills from older students as it has 2 separate components: a pin and a sleeve (designed for the OLLO PRO and DREAM systems). Similarly, please note that **ROBOTIS materials** refer to this smaller rivet as the **6 mm** rivet (6 mm being the distance between **adjacent** mounting holes).
- Fig. 1.2 also shows that the RB-86 and other KINDER frame parts have matching holes for both types of rivets.

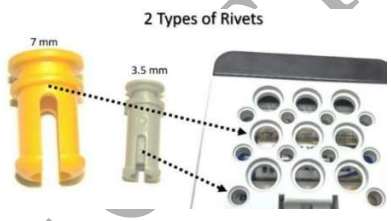

Fig. 1.2 ROBOTIS' rivet systems: 12 mm and 6 mm.

.....

1.2 ROBOTIS Software Design Approach

With the new OLLO Systems, ROBOTIS decided to go with a new approach to provide the users with the software tools needed to work with the new Controllers RB-86, RB-88, RB-100 and CM-151. This new approach is based on web services and is linked to the STEAMCUP Organization (https://enjoy.steamcup.org/). ROBOTIS recommends the use of the CHROME browser for best performance. The STEAMCUP App is also available via Google Play and Apple App Store for mobile devices. However, at present (3/2024), the STEAMCUP Android App does not work properly with Chromebooks.

1.2.1 STEAMCUP Overview

This STEAMCUP web site is an **international** gathering place where **each user** can **learn** to **develop** his/her OLLO projects, as well as **share** them via **CIRCLE**s and **POST**s. Most resources

are accessible to anonymous users, but some resources are accessible to registered members only, but **membership** is **free**! The reader is recommended to review these posts before proceeding further in this book:

- Introducing STEAMCUP (https://enjoy.steamcup.org/post/15532).
- Joining STEAMCUP & getting started with R-BLOCK (https://enjoy.steamcup.org/post/19891).
- Please note that the RB-86's firmware can only be updated via a registered account and the **MOBILE** STEAMCUP App. Furthermore, both RB-86 and RB-88 use the same firmware (currently V.20 in March 2024).
- If a Firmware Recovery/Update is needed, please review this post (https://enjoy.steamcup.org/post/23034).

Currently (2024), the OLLO software tools available are R-BLOCK, R-MOTION and CONTROLLER (which is a **virtual** version of the **RC-300** remote controller). Although any user can use these tools to develop an OLLO project, he/she will not be able to save their work onto the STEAMCUP's cloud drive, without creating and registering ONE account with STEAMCUP. However, anonymous users can EXPORT and IMPORT their work onto local PC drives. Furthermore, these software tools are strictly **one-to-one**, meaning that the user can only **connect** and **program** only **one** RB Controller via only **one** software tool, at any **one** time.

.....

1.2.2 Brief Tour of R-BLOCK

The author assumes that the reader has access to an RB-86 Controller and also has successfully connected to R-BLOCK (**on-line version**) directly via this link (https://rblock.steamcup.org/?lang=en). The reader can be an anonymous user for the purpose of Sub-Section 1.2.2.

After "canceling out" on R-BLOCK's question regarding "unsaved blocks", the user will need to choose the appropriate Controller, which is **RB-86** (see Fig. 1.10). Then the user will "continue" to be asked to choose between opening an "empty program" or "OLLO Kinder+" (which is the program already downloaded inside any RB-86 coming with the KINDER kit).

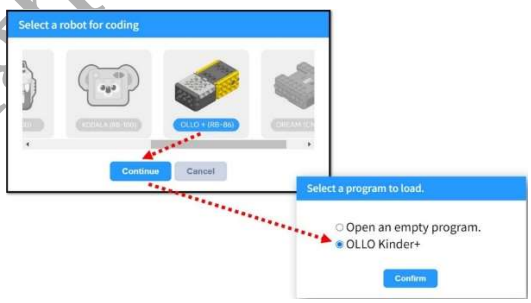

Fig. 1.10 Choosing Appropriate Controller and Opening Program.

Please choose the **OLLO Kinder+** program *(as of April 2024, the Off-Line version does not display and let the user access this file properly)*, and after a few seconds waiting for the STEAMCUP server to respond, the reader will see the contents of Fig. 1.11:

- The left side of Fig. 1.11 lists the various Block Groupings that are supported within the R-BLOCK's IDE for the RB-86.
- The Center Panel is reserved for the Main Program and its Functions (not shown in Fig. 1.11).

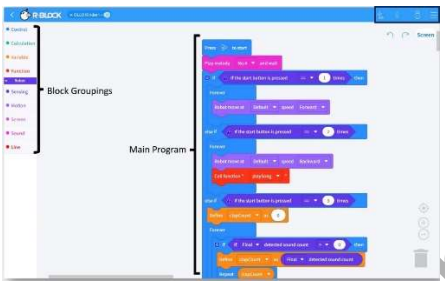

Fig. 1.11 R-BLOCK's Main User Interface.

Let's concentrate on the Main Toolbar found in the Right Upper corner of Fig. 1.11 whereas the reader can perform 3 actions (see Figs. 1.12 and 1.13):

- The **USB** icon is used when the user wants to connect to an RB-type Controller via a Wired USB Connection. This option is not available for the RB-86 and RB-88, but it is available for an RB-100.
- If the user clicks on the **Bluetooth** icon, a black window appears to allow the user to scan and to eventually connect to the built-in BT module inside the RB-86 (from the PC or Mobile Device). The user may see multiple controllers on this BT list, if he/she has powered on several RB controllers at the same time. However, the user can only connect to only ONE Controller at any ONE time, even though the STEAMCUP App may allow the user to instantiate several copies/windows of this App, in reality **the STEAMCUP Web Server effectively will allow ONLY a 1:1 session** per logged-in user. If the BT pairing executes successfully, the BT Icon will switch to its **connected** state (see Fig. 1.12).

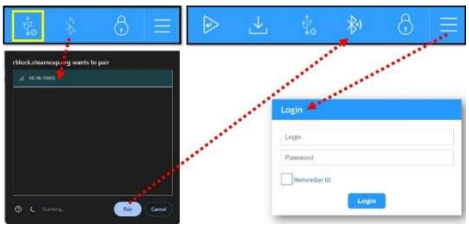

Fig. 1.12 R-BLOCK's Bluetooth Interface.

- Next, if the user clicks on the Hamburger Icon at the Top Right corner, two possible events would ensue:
    - If the user had **not logged in** yet into STEAMCUP, he/she would be asked to log in (Fig. 1.12).
    - If the user had **previously logged in**, then a new scroll menu would appear whereas the user can choose additional tools for the RB-86 (see Fig. 1.13). However, they will be described later in Chapter 18.

Fig. 1.13 Additional R-BLOCK's tools available via Hamburger Icon.

.....

In summary, Sub-Section 1.2.2 showcased how a completed R-BLOCK program for a KINDER robot would look like, but how did the original programmer arrive at this solution in the first place? This is a much more complex (and lengthy) subject/process to tackle. To start this process, the next Section 1.3 is a high-level overview of the main processes involved in creating an R-BLOCK solution to a given robotic project. More detailed step-by-step materials will be provided in Chapter 18 to "walk" the reader from project requirements through runtime analysis of the resulting R-BLOCK solution for an RB-86 (using several projects with increasing complexities).

1.3 Sense-Think-Act Paradigm

At the start of any robotic project, the programmer needs to create a set of goals and tasks that the robot would need to fulfill "physically" and "logically". Chapters 2 through 17 will help the reader with the "physical" issues, while Chapter 18 will help with the "logical/programming" issues.

If the robotic project is for a robotics competition, then these goals/tasks are likely pre-determined. But if the programmer is the one coming up with these goals/tasks, then it is "perfectly normal" for these tasks and goals to be "re-adjusted" throughout the duration of the project (i.e., a **trial-and-error** expectation is **OK**). Anyhow, regardless of the origin of the robot's goals and tasks, the **Sense-Think-Act (STA) Paradigm** will be a helpful approach to use for the user/programmer in obtaining a workable solution, and also in fine-tuning it for an ever-closer match between actual robot performances and its original task requirements.

.....

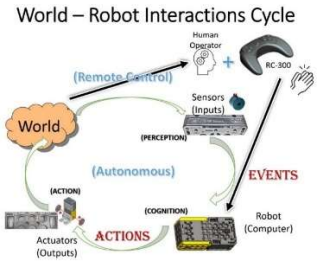

Fig. 1.20 World-Robot Interactions Cycle.

The Sense-Think-Act Paradigm will be revisited with more details in Chapter 18 to illustrate its adaptations to specific hardware/software environments used in different robot projects.

Chapter 5: Knight

The "Knight" project corresponds to Week 4 in the Kinder User Manual (pp. 14-17). This robot is powered and is of a 3-dimensional structure. It incorporates a 4-bar linkage mechanism on each side of the robot to manipulate a sword and shield.

5.1 Assembly Notes

Personally, I think that this robot design was a poor choice from ROBOTIS, as it was designed with its NIR sensors oriented towards the back side of the robot (see Fig. 5.1).

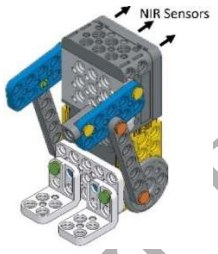

Fig. 5.1 ROBOTIS' Knight Robot.

Thus, the author modified this design into a Mod-Knight with the NIR sensors aimed forward instead (see Figs. 5.2 and 5.3).

Fig. 5.2 shows that Steps 1 through 3 in the User's Manual are followed to create the Basic Knight:

- The usual 1.5 rivet technique was used to fasten the two PL-2b2 to the PD-3b5 so the Basic Knight would not fall backwards (read more discussions in Section 5.3).
- All NIR sensors (1 through 6) are accessible for programming.
- The Basic Knight is used later in Chapter 18 to teach the basics of Robotics Programming.

Fig. 5.2 Basic Knight Robot.

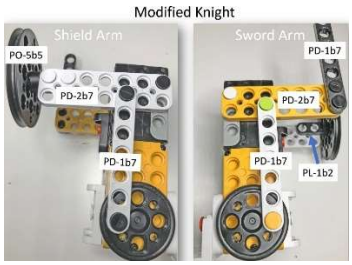

Fig. 5.3 Mod-Knight Robot.

Fig. 5.3 shows the assembly procedure for the Shield/Left Arm and the Sword/Right Arm:

- The Shield Arm was constructed from PD-1b7, PD-2b7, PL-1b2 (hidden behind PD-2b7) and PO-5b5 as a round shield.
- The Sword Arm was created from two PD-1b7s, and one PD-2b7.
- Because of the Shield, Mod-Knight can only "see" via NIR sensors #3 and #5 (see Fig. 5.4).

Fig. 5.4 Mod-Knight can only "see" via NIR sensor #3 and #5.

5.2 Mechanism Usage

A 4-bar Linkage mechanism is used for each arm of Mod-Knight as shown in Fig. 5.5.

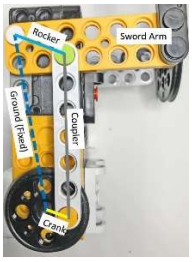

Fig. 5.5 4-Bar Linkage Mechanism used for Mod-Knight.

This 4-Bar linkage mechanism converts the smooth rotational motion of the internal motor to a raising/lowering action for the Shield Arm and Sword Arm:

- The CRANK (input link) is the Horn-5X which is powered by one of the built-in motors of the RB-86.
- The ROCKER (output link) is the Shield/Sword Arm.
- The COUPLER (intermediary link) connects the CRANK to the ROCKER. The Coupler's job is to transform the horn's rotational motion into the Rocker's waving motion.
- The GROUND link is in a vertical direction this time and it is embedded as usual in the overall body of the robot.

5.3 Motion Platform Features

As the Mod-Knight is motorized and can move around, we must also take note of its configuration features as a Motion Platform (see Fig. 5.6):

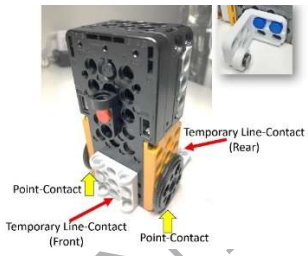

Fig. 5.6 Selected Features of "Mod-Knight" as a Motion Platform.

.....

5.4 Ideas for Further Explorations

5.4.1 Coordinating Shield and Sword Motions

Fig. 5.7 shows that the Shield Arm and the Sword Arm can be used in In-Phase or Out-Of-Phase (180 degrees).

.....

For the Mod-Knight, the Out-Of-Phase usage is the more "realistic" one as a "real" knight would move the Shield up for self-protection while striking down with the Sword as an offense move.

5.4.2 Varying Linkage Lengths

The reader is encouraged to vary the position of the Shoulder Joint (Coupler-Rocker Joint) as shown in Fig. 5.8. The Shoulder Joint can also be relocated to different places resulting in different ranges of motion for the Shield and Sword Arms. For some combination of positions, this mechanism could be jammed at spots within its normal trajectory.

5.4.3 Combining 4-Bar Linkages >> Boxer

Expanding on the Martial Arts theme in this project, we can combine a Parallel Motion Linkage to the existing 4-Bar Linkage to create a Punching motion and obtain a Boxer robot, using only the parts supplied with the KINDER kit (see Figs. 5.9 through 5.12).

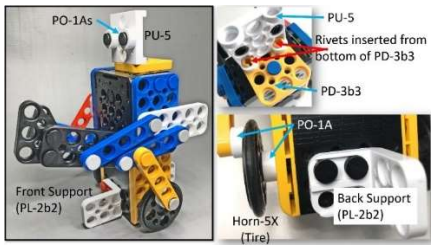

Fig. 5.9 Boxer Robot "Head" and "Leg" Assemblies.

.....

5.4.4 Looking Forward

The "Knight" theme is revisited in the EXCEL Kit as a "Jousting Knight" on a Horse and using an XL-330 actuator to control the motion of the Lance (see Fig. 5.14).

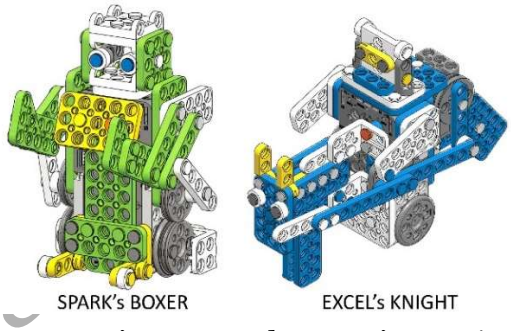

Fig. 5.14 SPARK's Boxer and EXCEL's Jousting Knight.

The SPARK's "Boxer" combines two 4-Bar Linkages to provide the Punching motion with the RB-88 using the Mid-Axle for the Arm Motion and the Bottom-Axle for moving the robot in different directions (see Fig. 5.14).

.....

Chapter 18: Using R-BLOCK

It is well known that Block Programming/Coding tools are well suited to the youngest beginners (5+) for computer programming, in general, and for robotics programming, in particular (https://childhood101.com/block-coding-websites-kids/, https://www.shaperobotics.com/block-based-programming/).

As the RB-86 does not support AI functionality such as Speech Generation/Recognition, Multi-media or Object Recognition via Video Camera, both **on-line** and **off-line** versions of R-BLOCK are suitable for use with the KINDER system (please review Sub-Section 1.2.1 if needed).

Chapter 18 is written for a complete beginner in robotics and computer programming, and we will be using the "Basic Knight" platform and its variance (Fig. 18.1) to learn how to use the R-BLOCK App. A "Spiral" learning approach (Bergin, 2012) was implemented, whereas the author demonstrated how to use the R-BLOCK tool with projects having increasing scope and difficulty.

18.1 Construction of Basic Knight Robot

The Knight robot is used for Week 4 in the Kinder User Manual (pp. 14-17). For the purpose of Chapter 18, the reader/user needs to perform only Assembly Steps 1, 2, and 3 – resulting in the "Basic Knight" platform shown in Fig. 18.1. The main advantage for "Basic Knight" is that all 6 NIR Sensors are fully accessible to the programmer.

Fig. 18.1 Basic Knight Robot used for R-BLOCK Programming.

We are now ready to learn how to use the R-BLOCK tool and most importantly how to learn basic Robotics Programming Concepts and Techniques using this Basic Knight platform.

Please keep this STEAMCUP's Circle named **R-Block Manua**l handy for reference if needed (https://enjoy.steamcup.org/circle/1837).

18.2 R-BLOCK Menu System (RB-86)

Fig. 18.2 depicts the Start-Up Window for R-BLOCK (Off-Line) in Windows 11 OS:

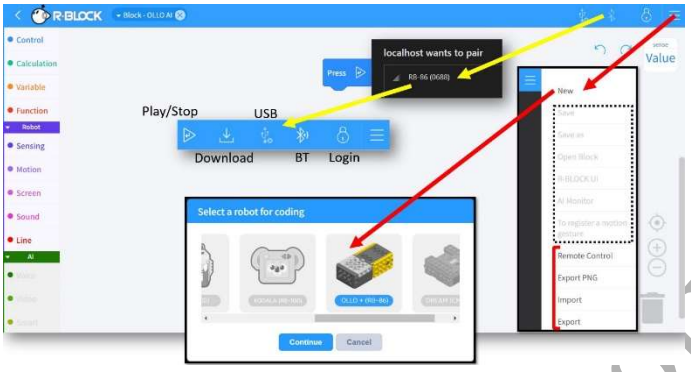

Fig. 18.2 R-BLOCK (Off-Line) Start Up Window.

- On the Left Edge, ones can see the various types of Programming/Coding Groupings that can be used in an R-BLOCK Program: **Control, Calculation, Variable, Function**, and Robot-related Groupings – **Sensing, Motion, Screen, Sound** and **Line**. Please note that the **AI** Grouping is disabled (grayed out) on the Off-Line version (for RB-86/88/100), but it would be enabled on RB-88/100 when using the On-Line version out of the STEAMCUP web site or mobile App.
- In the Top Right corner, ones can see 4 icons: USB, Bluetooth (BT), Padlock (Login), and Hamburger (3 horizontal lines). The RB-86 does not have a USB connection.
- The Hamburger icon allows the user to access a pull-down sub-menu with the following options: **New, Remote Control, Export PNG, Import,** and **Export**. Please note that there are several other options *grayed out*. Those will be *enabled* on the *On-Line* version of R-BLOCK, whereas the user has to log in to the STEAMCUP web site.

.....

18.2.1 CONTROL Blocks

Fig. 18.4 lists the actual Programming/Coding Blocks that are of the CONTROL type (Top to Down and Left to Right):

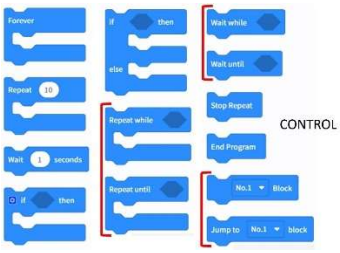

Fig. 18.4 CONTROL type of Programming Blocks for RB-86.

- FOREVER is an Endless Do Loop.
- REPEAT is a Counter-based Do Loop.
- WAIT puts the RB-86 into a "Do Nothing" state for a time period (seconds) specified by the programmer.
- Simple IF structure, based on specified Condition being evaluated to TRUE.
- IF-ELSE-IF structure, based on specified Condition. If Condition evaluates TRUE, the IF branch is executed. If Condition evaluates FALSE, the ELSE-IF branch is executed.
- Two REPEAT structures:
  - REPEAT WHILE executes as long as the specified Condition evaluates TRUE.
  - REPEAT UNTIL executes as long as the specified Condition evaluates FALSE.
- Two WAIT structures:
  - WAIT WHILE puts the RB-86 into a "Do Nothing" state as long as the Specified Condition remains TRUE.
  - WAIT UNTIL puts the RB-86 into a "Do Nothing" state as long as the Specified Condition remains FALSE.
- STOP REPEAT forces the Controller to terminate/exit the currently ACTIVE LOOP.
- END PROGRAM forces the Controller to terminate the currently running R-BLOCK program.
- The NUMBERED BLOCK element can be used to label the beginning of a special section of the R-BLOCK program.
- JUMP allows the Controller to immediately switch its execution task to the beginning of a user-defined NUMBERED BLOCK.

18.2.2 CALCULATION and VARIABLE Blocks

Fig. 18.5 shows the actual Programming/Coding Blocks that are of the CALCULATION and VARIABLE types. The CALCULATION types are described below (from the top):

.....

18.2.3 FUNCTION Blocks

Fig. 18.6 shows the actual Programming/Coding Blocks that are of the FUNCTION type, and they are described below (from the top):

.....

Please note that if the programmer creates several FUNCTIONs, they are all "stacked up" behind the most recently created FUNCTION.

18.2.4 SENSING Blocks

Fig. 18.7 lists the actual Programming/Coding Blocks that are of the SENSING type, and they are described below by sensor type:

.....

- If the user wants to implement Remote Control via the Virtual RC-300 tool, the programmer needs to use the following 3 Blocks:
  - REMOTE SIGNAL VALUE, IF THE REMOTE SIGNAL IS ..., and WAS THE RC SIGNAL RECEIVED.
  - More implementation details will be provided in later sections.
- When powering on the robot, if the user wants to associate the Number of Presses of the Power Button to trigger certain modes of operation for the robot, then the last 2 Blocks apply:
  - NUMBER OF TIMES START BUTTON IS PRESSED, and IF START BUTTON IS PRESSED ...
  - Fig. 1.18 shows an example usage.

18.2.5 MOTION Blocks

Fig. 18.8 illustrates the Programming/Coding Blocks that are of the MOTION type which are used to control the behavior of various Motors (Continuous-Turn and Position-Control modes) that can be attached to the RB-86.

In R-BLOCK, ROBOTIS implemented different **granularity levels** for the control of the 2 **built-in** Left and Right motors.

At the Beginner Level (Motion Control Level 1 – MCL1), the user can use the following Blocks:

- ROBOT GO FORWARD.
- ROBOT GO BACKWARD.
- ROBOT TURN LEFT.
- ROBOT TURN RIGHT.
- ROBOT STOP.

When these F/B/L/R Motion Blocks are executed, the robot (i.e. R-86 brick) moves in the Programmed **Direction** at a **Default Speed** and for a **Default Time Period** of 1 second, and then the robot **STOPS** – this is a very important "side-effect" that the programmer needs to be aware of. Furthermore, the default values for Motion Speed and Time cannot be changed by the user.

.....

18.2.6 SCREEN, SOUND and LINE Blocks

Fig. 18.10 displays the Programming/Coding Blocks that are of the SCREEN, SOUND and LINE types:

- SCREEN Blocks can only "print" **numerical** constants or variables, **without** or **with** a Line Break (i.e., goes to a new line below after printing).

18.3 Learning R-BLOCK Programming Basics

The author used R-BLOCK (V.2.0.1.6) to develop the programming materials demonstrated in this book.

The following sub-sections, 18.3.1 through 18.3.4, were written for a "complete beginner" reader in mind, but they could be used as "review materials" for readers with more experiences in Block Programming from elsewhere.

The reader is recommended to first view https://youtu.be/_SZy9PF7KU8 for a quick working tour of the R-BLOCK application (ON-LINE version) and to learn about the editing basics for R-BLOCK by following the author's steps in creating the example R-BLOCK project named "SequenceControl_0.rblock" (see Fig. 18.11).

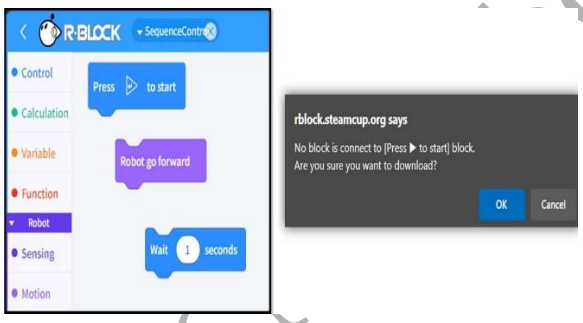

Fig. 18.11 Project "SequenceControl_0.rblock" and Test 1.

At this point, if the reader had been watching the previous video and had been practicing along with this video, then the reader is ready to continue to Sub-Section 18.3.1. If not, please start R-BLOCK (Off-Line or On-Line) on your PC and load up the project named "SequenceControl_0.rblock" before going on to Sub-Section 18.3.1.

18.3.1 Sequence Control

In this unit, we are going to learn that by default the RB-86 Controller behaves **sequentially**, i.e., it can do only one thing at a time, starting from the beginning and going down the list of Blocks/Commands. First, we are going to perform a few "tests" with "SequenceControl_0.rblock":

- **Test 1** (Fig. 18.11) – Just click on the Play Icon button found in the Top-Right area of the current R-BLOCK Window (also see Fig. 18.2). Then the user should see a black notification window appearing on the screen saying that the user should connect at least one Coding Block to the Start Block to run this (and any other) R-BLOCK projects. Here, if readers have used MIT SCRATCH before, the author would understand disappointments from these users who are accustomed to being able to "run" each Coding Block separately and

instantaneously. But this feature is not available with R-BLOCK as the runtime code resides on the RB-86 and not on the host PC. If the user connects these 3 Blocks together as shown in Fig. 18.12, this program will then compile and download fine to the RB-86. Next, if the user clicks on the RUN R-BLOCK button, the robot will roll forward for about 1 second and stop.

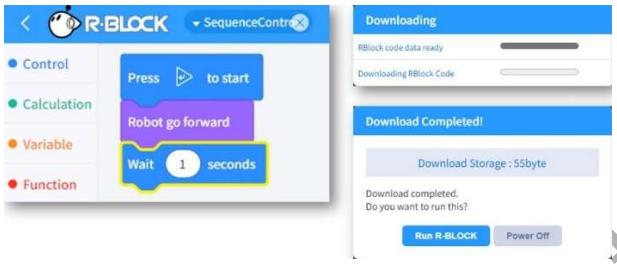

Fig. 18.12 Final Version of Test1 for Project "SequenceControl_0.rblock".

- **Test 2** (Fig. 18.13) –Let's just connect the "Music Scale" Block to **Start** and click on **Play** again. This time, we can hear the Buzzer sound out a "Low Do" note for 2 seconds, as expected.
- **Test 3** (Fig. 18.13) – Let's say that we now want the Buzzer to play "Melody 5", before playing the "Low Do" note. Thus, we connect the "Play Melody 5" Block above the previous "Music Scale" Block, as this would seem the logical step to do. Then, let's click on **Play** and we can hear Melody 5, but NO "Low Do" note at all. So, what is going on? Did we have "Robot Disobedience" already? In a way, we did:
    - When the Buzzer received the command from RB-86's MCU to play "Melody 5", it started to perform this assigned task right away. Now, back in the R-BLOCK program, RB-86's MCU was done with the "Play Melody 5" Block, so it went on to activate the next Block which was to "Play Note Do". As the RB-86's MCU is very fast, this was done within a few milliseconds from the previous command, and so the Buzzer did receive this new "Play Note Do" command, but it was still performing Melody 5 because a Melody lasts about 2 seconds. Consequently, the "Play Note Do" command was simply "lost/ignored" as the Buzzer was not designed to retain several consecutive commands at any one time. This is an important side-effect that an R-BLOCK programmer needs to account for, when there is a need to send different **consecutive commands to the same device** located on the RB-86. One simple solution is to use a "**Play Melody 5 and Wait"** command instead.

Fig. 18.13 Test 2 and Test 3 for Project "SequenceControl_0.rblock".

- As an extra exercise, the reader is recommended to put the "Play Note Do" command **before** the "Play Melody 5" command, and then recompile/reload this revised code. At runtime, the reader will find that Note Do is played for 2 seconds, then Melody 5 is next played in its entirety. This means that the RB-86's MCU would **"do nothing" during the execution of a "Play Note" command** and would not skip to the next command, until the specified Buzzer time delay has elapsed.

Let's next look at how **Motion Control Level 1 (MCL1)** Commands work with Sequence Control features of the RB-86's MCU (review Sub-Section 18.2.5 if needed).

.....

However, to create more complex maneuvers, stringing along these 1-second "Robot Go" Blocks is just awkward, editing-wise, and not computationally efficient either. In the next Sub-Section 18.3.2, we'll bring in the use of Motion Control Level 2 commands, along with Functions and Time Delays to design more efficient action/motion procedures.

18.3.2 Motion Control Level 2, Function & Time Delay

Fig. 18.16 shows the steps involved in creating and using Function Blocks:

1. Assuming that R-BLOCK IDE is running with a New Empty RB-86 workspace (see Fig. 18.2), the reader needs to switch to the Left Panel and choose **FUNCTION** Grouping.
2. Next, create 4 Functions named **Forward, Backward, Left,** and **Right**.
3. Drag a RUN FUNCTION Wrapper Block into the Editing Area.
4. Next choose **MOTION** Grouping from the Left Panel and drag out one Motion Control Level 2 (MCL2) Block named "Robot move at (Speed) (Direction)" into the Editing Area. Insert this "Robot move at ..." Block into the "Run Function Forward" Wrapper Structure (as shown in Fig. 18.16).
5. Then, go back to **FUNCTION** Grouping and drag a RETURN Block into the bottom of the "Run Function Forward" Wrapper Structure.
6. Lastly, insert a "Call Function Forward" Block (from FUNCTION Grouping) beneath the START Block. Thus, the MAIN BLOCK has only 1 command for this project "Forward_Jerk.rblock".

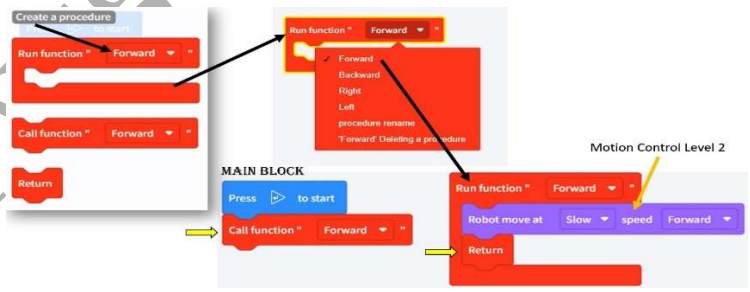

Fig. 18.16 Project "Forward_Jerk.rblock".

.....

The project “SequenceManeuvers_F.rblock” is a more complete application of the concepts presented earlier. Here, as an exercise, the author recommends that the reader does a “thought experiment” on this program before reading on the explanations for code sections A and B:

Fig. 18.18 Project “SequenceManeuvers_F.rblock”.

- For code section A: the robot turns right, then RB-86 does nothing for 1 second, then RB-86 makes the Buzzer play Melody 11 and waits here until the melody is finished (for a 2-second duration). So, in total, the robot turns right for ~3 seconds.
- For code section B: the robot rolls backward for 2 seconds, then RB-86 makes the Buzzer play Melody 10, but RB-86 keeps going to the STOP ROBOT Block. Therefore, the robot rolls backward only for 2+ seconds, and Melody 10 still plays while the robot is already stopped.
- For a more interesting behavior, the reader can try to put the “Play Melody 10” Block before the “Call Function Backward” Block and reduces the Wait Time to about 1.5 seconds, then the robot’s **motion and music play are “practically” synchronized from start to end**!

.....

18.4 Beginner’s Approach for R-BLOCK Projects

In this section, the author strives to provide beginning “roboteers” with a general approach to thinking about robotics projects, as well as concrete sequential steps that ultimately result in an efficient R-BLOCK program that can perform the tasks/behaviors that are required of the RB-86 robot for a given project.

18.4.1 Sense-Think-Act Paradigm (RB-86)

Fig. 18.23 represents the Sense-Think-Act (STA) Paradigm, first presented in Sub-Section 1.3, but now applied to the KINDER/RB-86 kit. STA will be used as a General Thinking Tool for solving various robotics projects from this point forward.

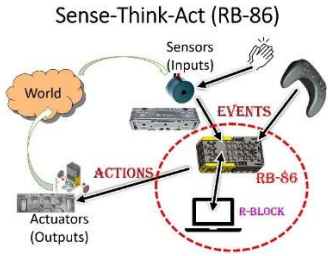

Fig. 18.23 Sense-Think-Act Paradigm as applied to RB-86.

We will be using the "Sense-Think-Act" Paradigm to help us keep the "big picture" of any robotics project in mind, as we start our "drilling down" journey into this paradigm to get to the actual robot-controlling code that works on the R-BLOCK IDE. We also will be leveraging all the Programming Basics learned from the previous Section 18.3.

We will be developing two projects using this approach: one in Autonomous-Behavior mode ("Spinning_Top_2_XX.rblock"), and the other in Remote-Control mode ("Clapping_RC_X.rblock").

Next, let's spell out the goals/rules for a new project "Escaping Spinning Top" that will lead to several example codes, labeled as "Spinning_Top_2_XX.rblock":

- The Average Value of NIR Sensors #1 and #2 will be used to detect how "far" a blocking object is from it.
- If no obstacle is detected, the robot should stay still.
- If an obstacle is detected "far", the robot should spin left at slow speed.
- If an obstacle is detected "mid-range", the robot should spin left at default speed.
- If an obstacle is detected "close", the robot should run away, i.e., go forward at high speed.

18.4.2 Reactive Control

Let's see how we can use the "Sense-Think-Act" (STA) Paradigm to help us program the KINDER/RB-86 robot to perform the previous set of tasks. We'll be concentrating on the dashed red boundary in the lower right corner of Fig. 18.23 (i.e., zooming in on the Computing Hardware components).

STA still applies at this level, but different robotics practitioners would start using different labels such as "Events", "Actions" and "Reactive Control" as they are more useful in the next steps that we need to take to create the actual computer code to control the KINDER robot. Interested readers should read Matarić (2007) for a more complete description of other robotics control approaches, or Ben-Ari and Mondada (2018) for more details about "Reactive Control" and "Finite State Machines".

18.4.3 Event-Action Pairs

With the "Reactive Control" (RC) approach, the first step RC1 is to translate each project goal from the above list into a specific Condition/Event matched with an appropriate Action for the robot to do (see Fig. 18.24).

Reactive Control Approach

Given Condition (Event) >> Appropriate Robot Action

Condition/Event	Action
No Obstacle	Robot Stays Still
Obstacle Detected "Far"	Robot Spins Left @ Slow Speed
Obstacle Detected "Mid-Range"	Robot Spins Left @ Default Speed
Obstacle Detected "Close"	Robot Forward @ Fast Speed

Fig. 18.24 Event-Action Pairs.

At this stage of the process, the reader needs not consider any hardware or software solutions in mind at all and just performs a "thought experiment" or "visualization exercise" whereas the reader would imagine to be "one" with the robot and wait for a specific event to happen, and then would perform the appropriate maneuver.

Before we go on to the next step, a closer look at these four Event-Action pairs listed in Fig. 18.24 should show that they are mutually exclusive, meaning that at run time, one and only one of these Event-Action Pair can be active at any one time.

18.4.4 Sensor-Actuator Pairs

The second step RC2 is to build upon the results of the previous step RC1 by mirroring its Event-Action pair into a matching Sensor-Actuator pair as shown in Fig. 18.25. The goal is to specify the kind of Sensors/Actuators to be used and to prescribe how they are to be used.

Input-Output Table

Given Input Sensor(s) >> Activate Appro. Actuator(s)

Input Sensors	Output Actuators
Average IR <=50	Robot Stop (MCL1)
Average IR between 50 & 200	Robot Turns Left Slow (MCL2)
Average IR between 200 & 350	Robot Turns Left Default (MCL2)
Average IR > 350	Robot Forward Fast (MCL2)

Fig. 18.25 Sensor-Actuator Pairs corresponding to Event-Action Pairs listed in Fig. 18.24.

For this step, the reader can see that we need to apply the Programming Basics learned from Section 18.3 to "translate" the "general" definitions of "No Obstacle", "Object Detected Far", "Object Detected Mid-Range" and "Object Detected Close" into "concrete" values for the Average IR parameter that are used in the "Input Sensors" column of Fig. 18.25.

.....

In the "Computational Thinking" practice, the previous steps would correspond to the "Decomposition" phase (Beecher, 2017; Denning and Tedre, 2019).

18.5 Remote Control Options

There are two basic requirements of any remote-control scheme:

1. Quick and accurate robot's compliance to a command from the human operator.
2. When there is no command from the operator or if a loss of communications is detected, the robot needs to set itself into a "safe" configuration – safe for the robot as well safe for the human operator.

For the KINDER/RB-86 kit, there are two ways to perform remote operation of the robot: one is through its built-in buzzer/microphone by clapping hands, the other is by sending it messages via its built-in Bluetooth receiver.

18.5.1 Clap based Remote Control

First, let's spell out the goals/rules for a new project "Clapping Remote-Control" that can lead to two demo programs, labeled as "Clapping_RC_1.rblock" and "Clapping_RC_2.rblock":

- The RB-86 Buzzer/Microphone will be used to detect the number of hand claps (or loud noises) originating from the human operator to serve as a Remote-Control signal for the KINDER robot.
- If no hand clap is detected, the robot should stay still.
- If ONE hand clap is detected, the robot should Go Forward.
- If TWO hand claps are detected, the robot should Go Backward.
- If THREE hand claps are detected, the robot should Turn Left.
- If FOUR hand claps are detected, the robot should Turn Right.

18.5.2 RC-300 based Remote Control

Although the Physical RC-300 Remote Controller is not included in the KINDER kit, the Virtual RC-300 Remote Controller is available with the R-BLOCK IDE (see Fig. 18.36). The Virtual RC-300 is a graphical panel mimicking the Physical RC-300 thus it has all 10 Buttons U/D/L/R/1/2/3/4/5/6.

Fig. 18.36 Virtual RC-300 Remote Controller from R-BLOCK IDE.

Unlike "Clap-based Control" which has only one way of taking command inputs, i.e. via hand claps, the Virtual RC-300 device has two ways of activating its 10 Buttons:

1. The Virtual RC-300 can be operated with the Mouse's Left Button, thus the user can only "click/push" on **one** of the 10 buttons available on the Virtual RC-300 at **any one time**. On the other hand, the Physical RC-300 is designed such that the user can push on multiple buttons of the same 10 buttons **at once**.
2. To overcome the previous limitation, the Virtual RC-300 is also designed to take command inputs via the standard **keyboard**, namely via the U/D/L/R **Arrow keys** and the **Number keys** on the keyboard's top row. Thus, the operator now has the option of pushing on 1 key or multiple keys just like on the Physical RC-300.

First, we'll use the R-BLOCK program "RC300.rblock" to demonstrate and understand the similarities and differences between the Virtual and Physical RC-300 devices and when only **one button is being activated**.

.....

18.6 Object Detection & Line Tracking

This Section showcases R-BLOCK projects that require **Fast Sensor Inputs** and **Responsive Motor Actions**. We'll use the same "Basic Knight" robot shown in Fig 18.1, and it will be using the **Current Values** of the NIR Sensors out of Port 6 (Left_IR) and Port 5 (Right_IR) – see Fig. 18.44.

Fig. 18.44 Function "Read_IR_Sensors" used for the Avoider projects.

In Function "Read_IR_Sensors", the multiplication factor of "1" is just there for the reader's convenience in case the reader needs to use a multiplication factor different than 1.

18.6.1 Avoider

The "Avoider" will be using its Left (Port 6) and Right (Port 5) NIR Sensors to detect a sin-gle object and to accomplish the tasks described in the following Event-Action list:

- If there is no obstacle in front of Avoider, it can go forward.
- If there is an object on Avoider's Left, it should back away and turn to the Right to avoid the obstacle.

- If there is an object on Avoider's Right, it should back away and turn to the Left to avoid the obstacle.
- If there is an object in front of Avoider, it should go straight Back to avoid the obstacle.

18.6.2 Smart Avoider

For the "Smart Avoider" Project, We'll use the same "Basic Knight" robot shown in Fig 18.1. The "Smart Avoider" will be using its Left (Port 6) and Right (Port 5) NIR Sensors to detect a single object.

The "Smart Avoider" Project shows how to combine Autonomous-Behavior and Remote-Control techniques so that the Avoider robot can "save itself" when, on purpose, the user drives the robot straight into an obstacle. Essentially, we'll try to program a little bit of "intelligence" into the robot! The resulting program is called "RC300_SmartAvoider.rblock": it combines codes from "RC300_DB.rblock" and codes from the Avoider project.

.....

This "Escape" algorithm was designed to respond to a single obstacle such as a flat wall or fence, but the reader can try to send this robot into a wall/fence corner to see how the robot would maneuver itself then. The author had found that most of time this algorithm would handle a corner well, but there are instances where it would get "stuck" bouncing back and forth between the two walls/fences.

As a comparison, the reader may check out the equivalent "Smart Avoider" solution using MIT SCRATCH-2 running on a Desktop PC (https://www.youtube.com/watch?v=o8w_YVocCpA)

18.6.3 Line Trackers

Back in Sub-Section 18.2.6, the author mentioned that R-BLOCK offers new functionalities in the LINES Block such as the capability to find different types of track intersections while the robot is rolling forward (see Fig. 18.59).

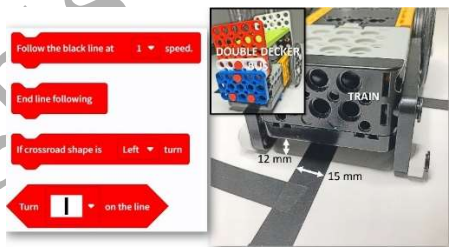

Fig. 18.59 New LINES Blocks from R-BLOCK tool.

18.7 Position Control of Servo Motors

The last project of this book is to show the reader a basic usage of an external actuator such as the XL-330 (see Fig. 18.67):

- The XL-330 does not come with the KINDER kit, so it needs to be purchased at various on-line shops such as the ROBOTIS e-Shop (https://www.robotis.us/dynamixel-xl330-m288-t/).
- We'll be using Dual-Lock tape to attach an XL-330 "stably enough" to the robot platform (https://www.amazon.com/dp/B07XMGCSM6/).
- We'll be using a simple arm to knock away any frontal obstacle that the robot may find as it is tracking an arbitrary black line (like in the previous project).

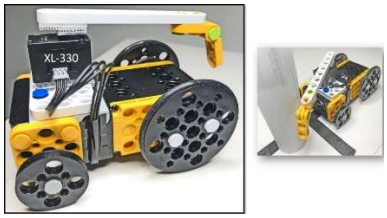

Fig. 18.67 Line-Tracker with Object Detection and Object Clearing using an XL-330.

For this project "Line_Tracker_OD.rblock", we'll be using the Frontal NIR Sensors #1 and #2 to help the robot detect an object in front of it, as it is doing its main job of tracking a black line, just like in the previous project (see Fig. 18.68):

18.8 Standards Alignment

Georgia Standards for Excellence for K-8 Computer Science:

- CSS-EL.K-2.1: Empowered Learner.
- CSS.KC.K-2.2: Knowledge Constructor.
- CSS.IDC.K-2.4: Innovative Designer and Creator.
- CSS.CT.K-2.5: Computational Thinker.