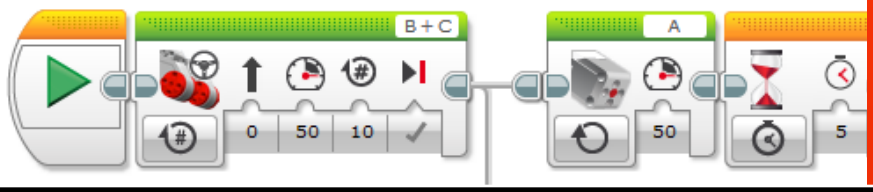
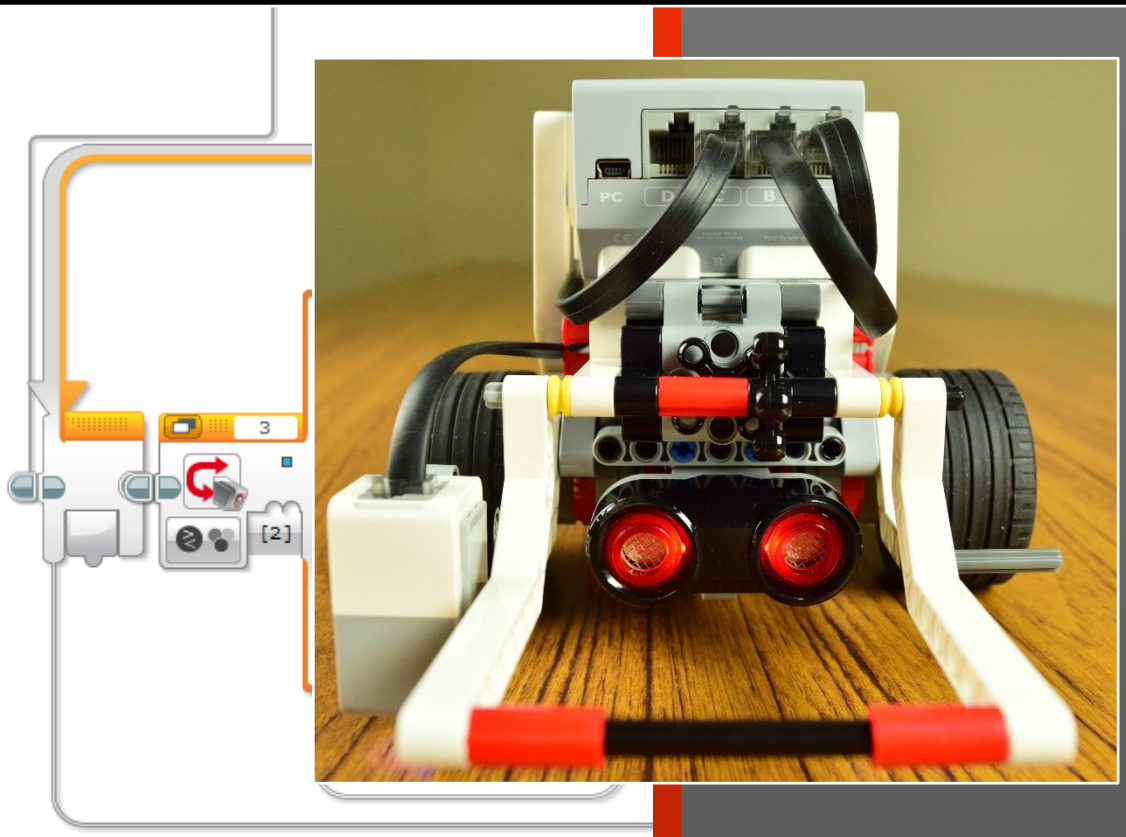
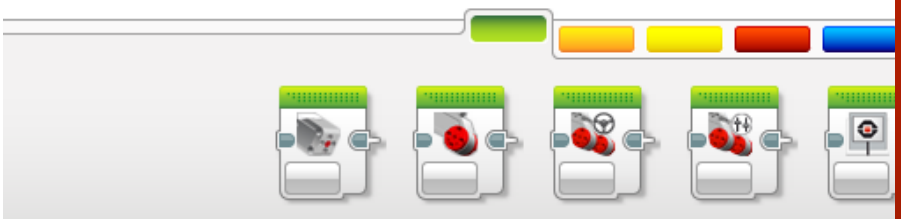


2016

LEGO Mindstorms EV3 Programming Basics





Joe Olayvar & Evelyn Lindberg

Washington State Library

Library Development Team

SOS Office of the Secretary of State Washington State Library

Institute of Museum and Library Services

Table of Contents

About This Tutorial:

This two-phased tutorial of written and video instruction is completely self-paced with lots of opportunity to pause, absorb, and try things out on your own. When a video is available for a module, it is noted by "[Video #]" in the heading so that you may quickly queue it up from the CD. When you get to the exercises, however, the idea is that you first attempt the exercise before watching the related video. Everything you need to successfully complete the exercises is addressed in the curriculum and instructional videos. The exercise video's themselves have no narration, but do have the occasional tip.

Though geared for the beginner, this tutorial could also be used as a refresher. The only requirement is having a LEGO Education Core Set to build the robot, and the EV3 Software installed on a computer. Or, be a library participating in the Gaining STEAM Lego Mindstorms Circulation Kits provided by the Washington State Library and the Institute of Museum and Library Services. If that be the case, everything you'll need is on hand.

The Lego Mindstorms Robotics system, which includes the EV3 Programming Software, can be as advanced or as basic as you'd like it to be. But for our purposes, we'll just be covering the essentials of the system to demystify some programming concepts and set the foundation for building the imagination and creativity that are fundamental to Science, Technology, Engineering, Arts, and Math; also known as STEAM.

Through this introduction into programming and robotics, you will learn the thought process behind creating a program, basic programming functions, and how they relate to robotic actions and reactions. The curriculum itself is broken down into modules with most having accompanying videos. In all, there are 22 videos ranging from under one minute, to around 18 minutes, with a total run time of roughly 1½ hours.

So whether you just do specific modules as a refresher, or start at the top as a beginner, it is completely up to you. But whatever you choose, remember to let your inner kid come out to play, experiment, and most of all, HAVE FUN!

Preparing For This Tutorial:

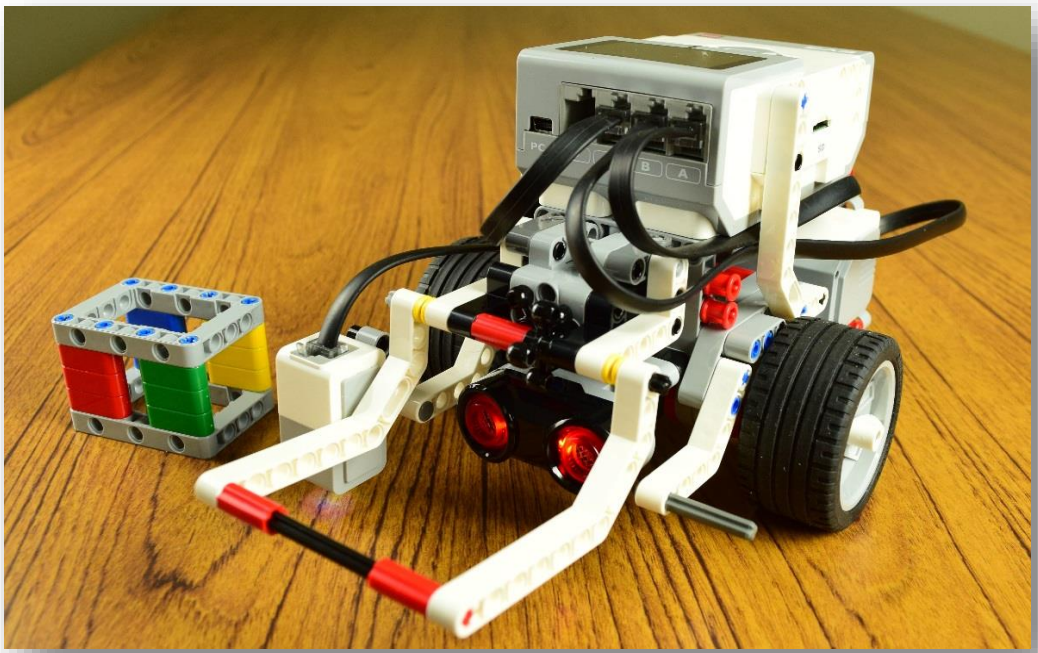
The LEGO Mindstorm EV3 Robot that coincides with this tutorial comes from building specific sections found in the LEGO Mindstorm Education Core Set building instructions.

You will need to build the main body for the robot (I'll refer to as the Base Unit), plus two sensor assemblies and a Medium Motor Arm assembly that mount onto the Base Unit. Not everything in the LEGO instructions is needed for this robot, so *only build the following assemblies and mount them to the Base Unit per the instructions*.

Here are the specific assemblies and the pages that outline their construction:

Assembly	**Pages**
Color Cube	4 thru 6
Base Unit	7 thru 38
Ultrasonic Sensor	42 thru 46
Medium Motor Arm	54 thru 67
Color Sensor	69 thru 71

When complete, you should have a robot that looks *exactly* like this

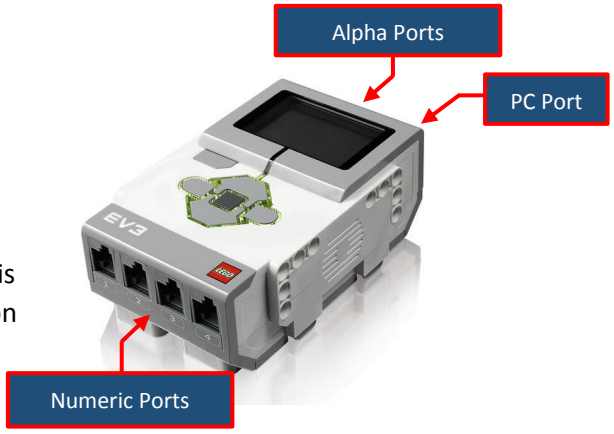


You will also need:

1. A flat and smooth surface area of at least 3' X 3' (or close to it) for the robot to run its programs freely.
    a. Most of the exercises can be confined to a smaller area and some can even run while connected to the computer, so the average portable table top could work.
    b. If floor space is your only option, a smooth floor is best, but a very low and tight-threaded carpet will suffice for most exercises.
        i. Carpet will not work very well for any exercises using the Color Sensor.
    c. Having both floor types available is a great way to test/illustrate movement variables.
2. Electrical tape (or something similar) of at least one basic color that will contrast with the surface area.
    a. Tan masking tape will NOT work (inconsistent color readings), but the blue or green should.
    b. Having multiple colors is a great way to test variables and possibilities.
3. To charge the battery (does not apply if using AA's).
    a. In theory, the battery is already attached to the "Brick," (the robot's computer/control center) so it's just a matter of plugging the charger into it.
    b. A fully charged battery will display one green light while still plugged into an outlet. During charging there *may* be just a red light to start, but normally it's both a green and a red light. A full charge could take a few hours depending on its current state.

The main LEGO Mindstorm components used for our robot:

- **The computer/control center…a.k.a. the "Brick"**
    - Programs are executed from the Brick.
    - Programs are downloaded to, or created from the Brick.
        - NOTE: directly programming on the brick using its interface is possible, but a bit cumbersome, and will not be covered in this tutorial. There are examples of this function in the EV3 instruction manual for your reference.



    - Sends programmed information to motors.
    - Receives information from sensors to initiate programmed parameters.
    - Must be connected to Motors and Sensors via cabling in order to function.
    - Motors MUST be connected to the front alpha ports.
        - NOTE: alpha ports send information out.
    - Sensors MUST be connected to the rear numeric ports.
        - NOTE: numeric ports receive information in.
    - Plays programmed sounds and displays.
    - Connects to the computer via PC Port that's located near the Alpha Ports.
    - See "Brick Overview" on page 8 for more information.

- **Large Motor**
  - Receives programmed instructions from the Brick.
  - Heavy duty with lower gearing for mobility requirements.

- **Medium Motor**
  - Receives programmed instructions from the Brick.
  - Used primarily for moving parts of a robot.
  - Light duty with higher gearing for quick response; NOT suitable for mobility.

- **Color Sensor**
  - Detects color differences and sends that data to the Brick for possible action according to programmed parameters.
  - Emits a set of color wavelengths. The color wavelength that is reflected back, and not absorbed, determines the color of the surface.
  - NOTE: sometimes, what appears to be a certain color to our eyes, may reflect differently to the sensor and cause unexpected results.

- **Ultrasonic Sensor**
  - Detects distance to an object and sends that data to the Brick for possible action according to programmed parameters.
  - Using the same principle as Bats and submarines, it uses echo location by emitting an ultrasonic wave that is received back after bouncing off an object. The time it takes to be received back determines the distance to that object.

A few other LEGO Mindstorm components…NOT used for our robot:

- **Touch Sensor**
  1. Sends the state of either being pressed, not pressed, or bumped (pressed, then not pressed) to the Brick for possible action according to programmed parameters.
  2. Often used to determine if the robot has physically bumped into something.

- **IR (infrared) Sensor** *and* **IR Beacon**
  1. The IR Sensor detects proximity to other objects much like the Ultrasonic Sensor and sends that data to the Brick for possible action according to programmed parameters.
  2. Also detects IR signals from the Beacon.
  3. Beacon can be used as a hand-held remote control.

- **Gyro Sensor**
  1. Detects the robots orientation and rotational motion and sends that data to the Brick for possible action according to programmed parameters.

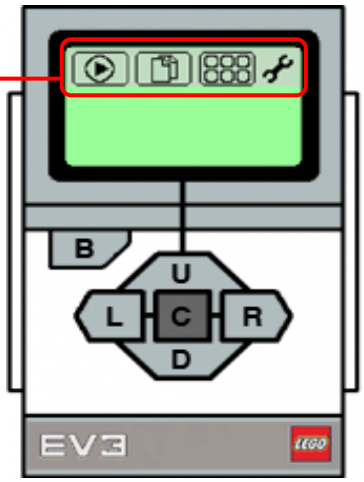
Brick Overview: [Video 01]

As mentioned earlier, the Brick is the control center of your robot's functionality. All your programs are downloaded to it and run from it. All your connected Motors are sent commands from it, and all your connected Sensors provide information to it for strict interpretation as applied to the program being run. Also mentioned is that Motors MUST be connected to the Alpha ports, and Sensors MUST be connected to the Numeric ports for them to function.

That's the basics of what the Brick does, so now let's learn how to turn it on, and navigate the interface.

NOTE: Button lettering is informational only; they are not actually on the Brick buttons.

The Buttons:

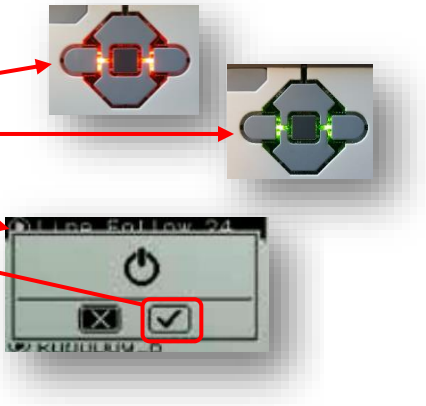


- [B]ack | Cancels Action, Aborts Running Program (important to remember), Navigates to previous screens and Shut Down screen.
- [C]enter | Turns the Brick On, Selects highlighted option
- [U]p | Navigates Up a list
- [D]own | Navigates Down a list
- [L]eft | Navigates Left across the Menu Tabs
- [R]ight | Navigates Right across the Menu Tabs

Menu Tab Icons:

- **Run Recent** | Lists all programs recently run
- **File Navigation** | Lists all Project Folders and the Programs within them that are loaded on the EV3
- **Brick Applications** | Provides access to native apps such as Port View (displays connected Motors and Sensors) and the Brick Program (enables programming directly using the interface) among other options.
- **Settings** | General configurations such as Volume, Sleep, and Wireless settings are made here

ON and OFF:



- **ON** | Press the [C]enter button until a red light comes on
  - Changes to green when ready for use
- **Off** | Press the [B]ack button until you see the Shut Off screen
  - Tab [R]ight to highlight the check mark
  - Press the [C]enter button to initiate shut down
  - The light will turn red, then off when fully shut down

Programming Workflow:

Basic Programming Workflow Model

1. Create Program (initial instructions / connected programming blocks)
2. Test/Run the Program
3. Experience the eternal question: "Why isn't it working?"
4. Modify Program
5. Rinse and Repeat steps 2 thru 4 as necessary

The "Basic Programming Workflow Model" (as outlined by Evelyn Lindberg) is the very essence of this tutorial. Though we will be taking things very slowly at first, the approach is the same throughout. Plus, the Final Challenge will put this workflow to good use, so I'd like to break it down a bit and apply it to Lego Mindstorms EV3 Programming.

1. **Create Program:** In EV3 terms, this means connecting Programming Blocks together that *we hope* work in concert with each other for a desired outcome. But part of that creation process is planning, or what might be considered as pseudo programming. For us, as we think about programming our robot to do specific tasks, it might look something like this.

    a. Follow a line to an obstacle.
    b. Stop at the obstacle and grab it.
    c. Backup in an arc then stop.
    d. Release the obstacle and make a victory noise.

    With that bit of planning applied to EV3 programming, we could figure out what our robot might need in terms of Motors and Sensors, how it's constructed, as well as what type of Programming Blocks should be used.

2. **Test/Run the Program:** As with all things that have function, programs need to be tested. What's nice about the EV3 programming method is the ease in which you can test programs and portions of programs. This is due to how EV3 programs execute; starting with the first block, each block executes, then hands control over to the next block in line…and so on. Visually, this provides a means to "see" blocks being executed; on the programs canvas (if your bot is connected), or by the specific moves of your robot that correlates with each block. There are slight exceptions to this run-first scenario, but for the most part, it holds true.

3. **Experience the eternal question: "Why isn't it working":** It's fairly rare that a program of any length runs correctly the very first time. It may *effectively* run correctly with all the correct robotic moves, but not necessarily in ways intended, or even expected. A myriad of things can be slightly off to make your robot be, well, slightly off. Any result other than exactly what you had in mind, needs debugging, or tweaking.

4. **Modify Program:** This is something you come to expect and in many ways, it can be the fun part…or not. But as mentioned earlier, the EV3 programming provides several clues as to where that undesirable outcome occurred in the programming. It's just a matter of recognizing the signs and modifying where needed.

5. **Rinse and Repeat steps 2 thru 4 as necessary:** Eventually, victory will reign and you'll be able to skip this step #5. But until your robots programming works exactly as intended, Rinse and Repeat is the name of the game.

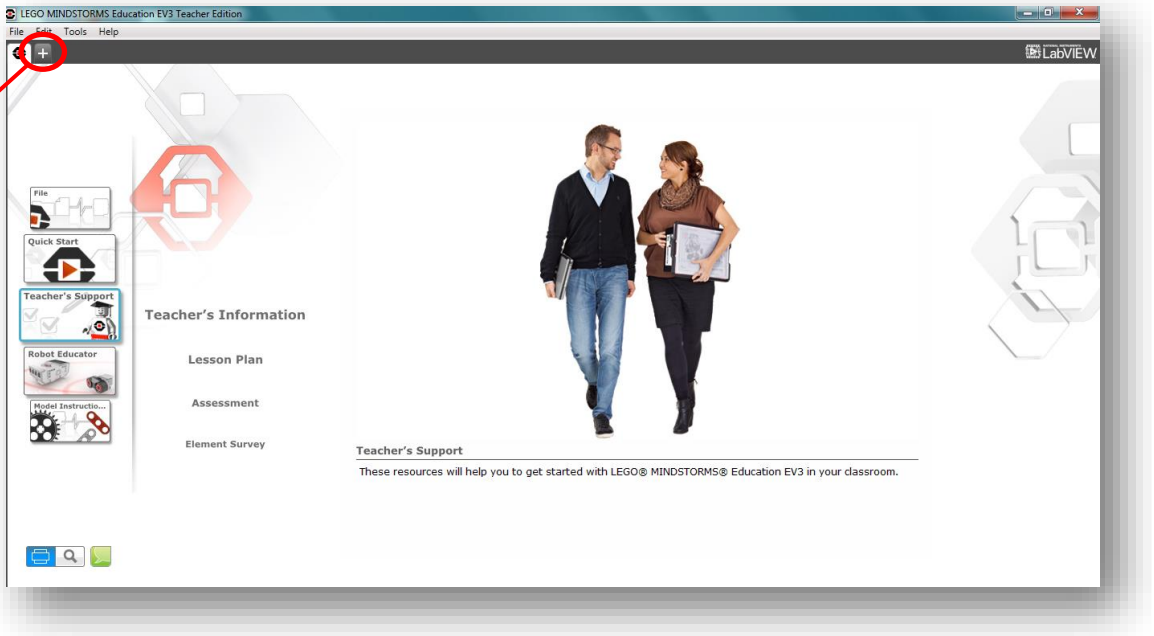
NOTE: As you progress through the programming exercises that start on page 16, I encourage you to explore block configurations to gain a better feel for the variables that are possible. Experimentation's a great way to learn.

Launching The EV3 Program: [Video 02]

On your desktop, find the EV3 icon -  . Double click it to launch into the EV3 Lobby Page.

There's a lot to be discussed about the Lobby and all that it offers. So much so, that I'll cover it in the video rather than attempt it here. But one thing I would like to show here is the quick way to get to the Programming Page.

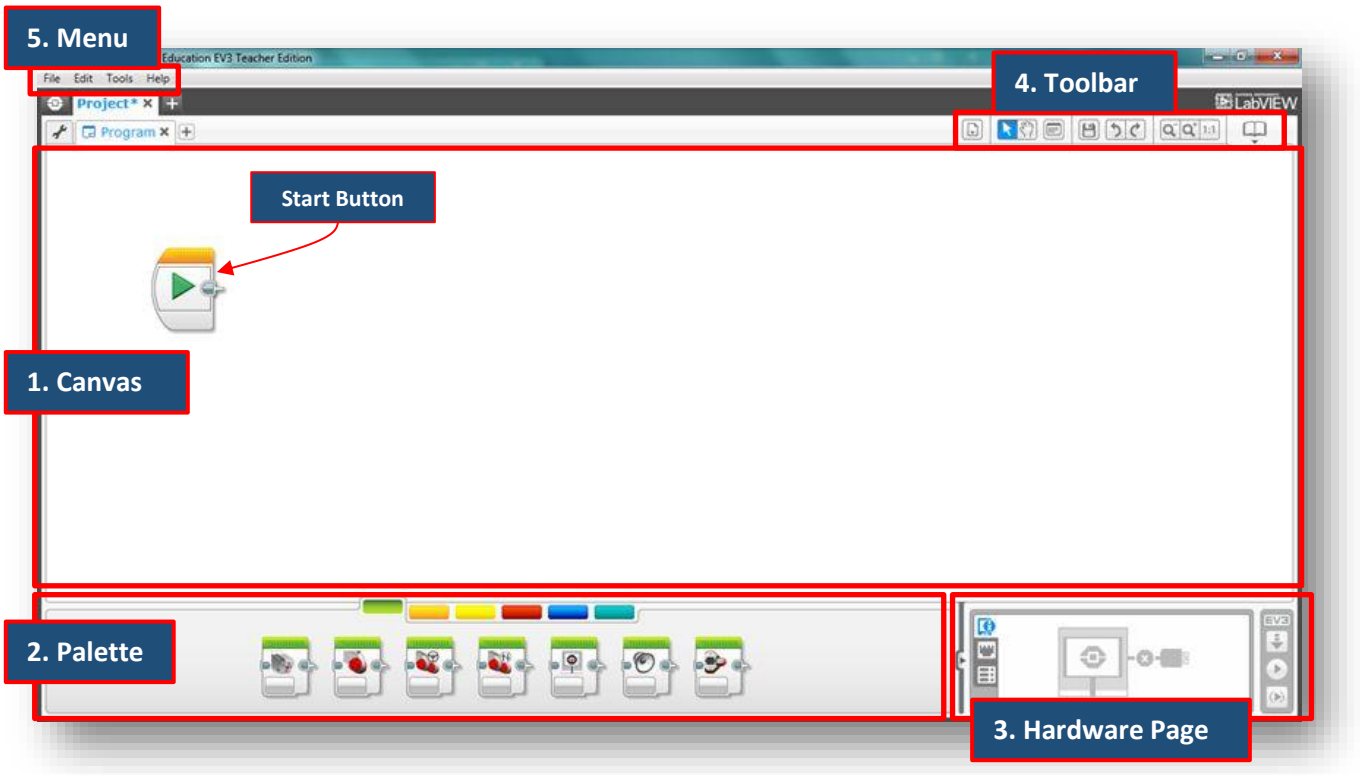
The easiest way to get to where everything happens, is to click the plus (+) sign in the upper left corner.



This will open the Programming Page that's discussed in the next section "EV3 Programming Screen Overview".

Besides several other items of interest, the video portion will discuss other ways to open new and existing Projects, as well as explore key "Helps" areas.

EV3 Programming Screen Overview: [Video 03]



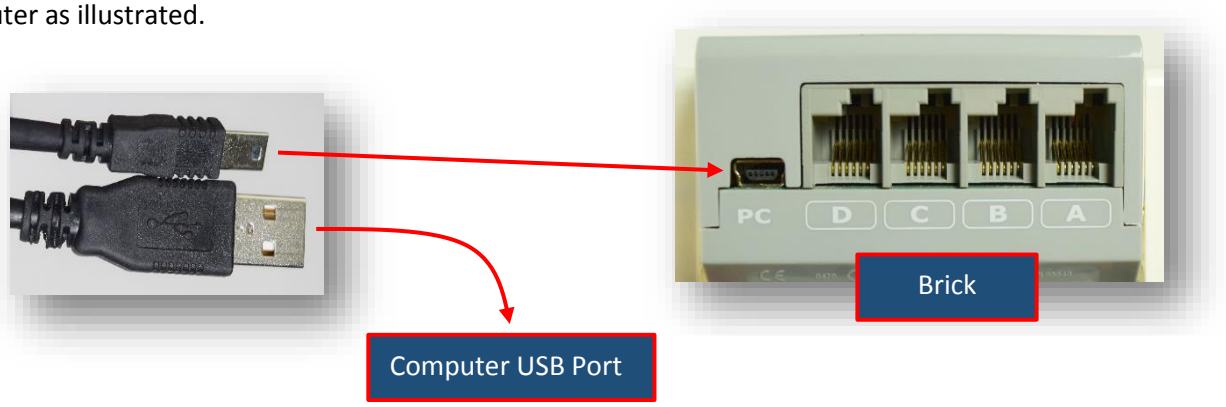
1. **Canvas**
    a. Programming area
        i. Create your program here by connecting Programming Blocks from the **Palette**
        ii. By default, a Flow Block Start Button is placed in the **Canvas** area
2. **Palette**
    a. Connectable Programming "Blocks" are located here *(hovering over blocks reveals type)*
    b. Each color has a function, but we will only focus on the Green and Orange in this tutorial.
        i. Green: Action
        ii. Orange: Flow Control
        iii. Yellow: Sensors
        iv. Red: Data Operations
        v. Blue: Advanced
        vi. Turquois: My Blocks
    c. Green Action Blocks
        i. Enables programmed movement, sound, and display capabilities
    d. Orange Flow Control Blocks *(I'll refer to as just Flow Blocks)*
        i. Modifies, or enhances the capabilities of other blocks
        ii. Cannot be used solo
        iii. For our purposes, we'll start with incorporating Green Action Blocks
        iv. As we progress, we will also combine Flow Blocks
3. **Hardware Page**
    a. Brick information displayed here
    b. Download center
        i. See "Downloading to the Brick / Hardware Page Details" (page 12) for more information
4. **Toolbar**
    a. **Canvas** area controls

5. **Menu**
    a. Very similar to feel and function of a Microsoft Menu Bar
    b. The "Show EV3 Help" (quick launch = F1) from the "Help" dropdown is excellent. It's one of the best Help links I've ever seen. Strongly suggest exploring this feature

NOTE: Reference the "EV3 Software Screens" document found on the CD for more Programming Page details.

Connecting Your Robot:

Besides the Alpha and Numeric Ports mentioned earlier, there are **PC**, USB, and SD Ports on the Brick as well. The SD Port is for expanding the memory capability and the USB port is actually for connecting Bricks together in sequence (daisy chaining), not for connecting to the Brick to the computer; the **PC Port** is how we do that. Using the USB cable that came with your Core Set, connect the small mini B end to the Brick's **PC Port**, and the standard end to your computer as illustrated.



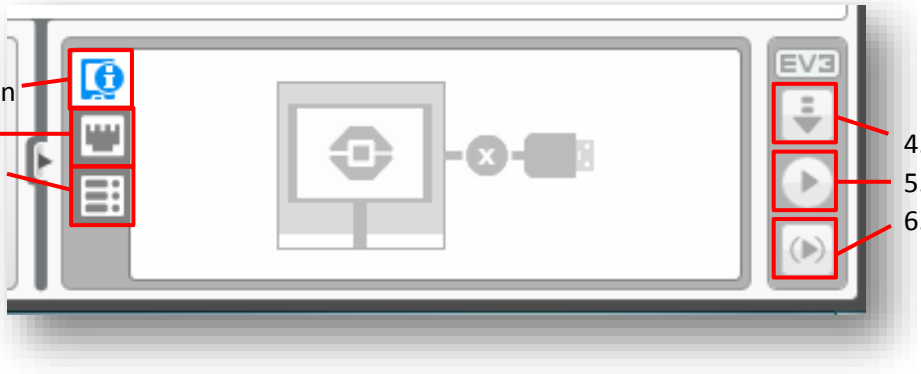
Downloading to the Brick / Hardware Page Details: [Video 04 *includes follow along programming*]

NOTE: In order to run your robot (run your program) untethered, you must first download the program completely onto the Brick. The Hardware Page is where you do this, but the Brick MUST be turned on (see "Brick Overview" on page 8) and connected (see "Connecting Your Robot" above) to the computer as well. To run your robot untethered, see "Running Untethered" on page 14.

On the left side of the Hardware Page are the tab selections "Brick Information," "Port View," and "Available Bricks." The "Available Bricks" tab is mostly for advanced use (daisy chaining) that we won't be covering. The "Brick Information" will be covered more in-depth further down the road as it pertains (for our use) to viewing and clearing program information. For the most part, "Port View" is where we will hang out when working with our connected robots.

On the right side of the Hardware Page are three buttons that provide "Download", "Download and Run", or "Run Selected" operations. Depending on where you are with your programming will determine which of these buttons is most useful at the time.

Here's all the previously mentioned tabs and buttons plus a little more Hardware Page info:



1. **Brick Information:** With the Brick turned on and connected, selecting this tab provides general information such as Firmware Version (basically the Operating System), Battery Level, Name (yes, you can name your bot here), as well as several options to manage settings and program information. Later, in the Brick Maintenance section, we will cover some of these options since by then we'll have some programs downloaded to view and talk about.

2. **Port View:** With the Brick turned on and connected, selecting this tab provides real-time information on the status of Motors and Sensors that are connected to the Brick. When creating programs, this is an invaluable tab to have displayed. Primarily, for verifying Port assignments and checking Sensor accuracy.

3. **Available Bricks:** For advanced use that we won't be covering in this basics tutorial.

4. **Download:** With the Brick turned on and connected, clicking this button will download your program to the Brick for non-connected operation. More on how to access and run a downloaded program in the following section "Running Untethered."

5. **Download and Run:** With the Brick turned on and connected, clicking this button will both download your program to the Brick, and run it at the same time. So be sure to have the space for your robot to do its thing, otherwise it may run off the table or knock your coffee over.

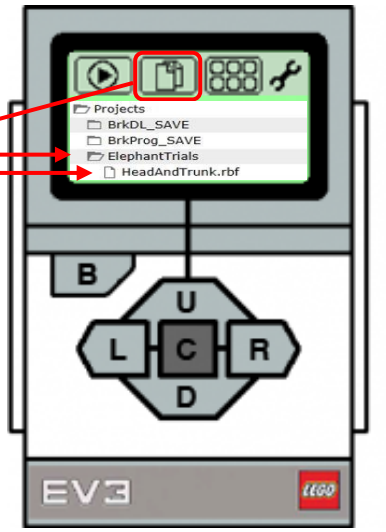
6. **Run Selected:** With the Brick turned on and connected, clicking this button will run a specified (highlighted) block, or string of blocks, as well as download to the Brick. This is a very useful tool. Basically, there will come a time that you want to trouble shoot, or simply test a certain outcome of a few blocks. By highlighting a set of connected blocks (even if they are part of a larger string), then clicking the "Run Selected" button, only those highlighted blocks will actually be run. This is also great when dealing with the sometimes troublesome "Sounds" or "Displays." When using the Sound or Display Blocks, occasionally your first attempt to download and run will produce no results on your Brick. Highlighting the Block and clicking the "Run Selected" button will oftentimes fix this anomaly.

Running Untethered: [Video 05]

Once you've downloaded a program to your Brick and have unplugged the USB cable, you need to find, select, and run the program directly from the Brick; a.k.a. run untethered.

In a nutshell, you navigate to "File Navigation" on the menu tab and drill down to your Project, then Program, and select it. More specifically, using the example at right, we will navigate to the Project "ElephantTrials" to select and run the Program "HeadAndTrunk"…

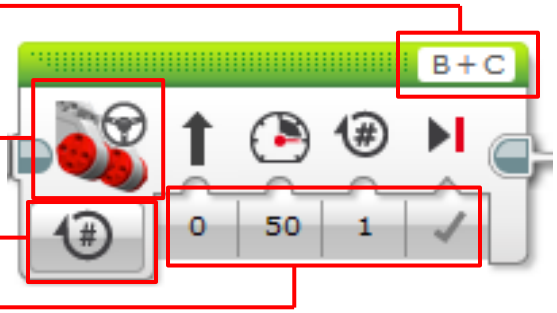
1. Depending on your starting point, press the [L]eft or [R]ight buttons until you're on the File Navigation Tab
2. Press the [D]own button to highlight your Project
3. Press the [C]enter button to select and reveal the Programs in that Project
4. Press the [D]own button to highlight your Program
5. Press the **[C]**enter button to select and launch your robot into action



Green Action Block Basics: [Video 06 *includes follow along programming*]

NOTE: Each block has unique characteristics, so the following focuses only on some common components, which themselves will vary. The intention is to gain a foundational understanding of the Action Blocks we will be using.

- **The Green Action Block**
  1. **Port Selector**
  2. **Block Type**
  3. **Mode**
  4. **Input Values**



- **In the above example**
  1. The **Port Selector** indicates what port that particular block has been assigned. By default, every motor block has an alpha port assigned. This can be changed as needed by selecting the field and choosing the actual alpha port a particular motor is physically connected to on the Brick. It will also change automatically (auto detect) if you have your robot turned on and connected to the computer prior to connecting blocks on the canvas.
     - Keep in mind that only Motor and Sensor Blocks have **Port Selector** fields since they represent a physical Motor or Sensor that needs to communicate with the Brick via cabling.
       - On the Brick,
         - Alpha ports <u>send</u> information to Motors
         - Numeric Ports <u>receive</u> information from Sensors
     - Also, rather than a **Port Selector** field, some blocks have a **File Name Input** field in the same location

IMPORTANT: Though you'll not likely have a port selection issue during our exercises, sooner or later…YOU WILL! Port selection is critical for successful operation of Motors and Sensors. Always verify that the ports assigned on your program block matches the physical ports your Motors *and Sensors* are connected to on the Brick itself. And remember, Motors must be physically connected to Alpha (send) ports, and Sensors to Numeric (receive) ports.

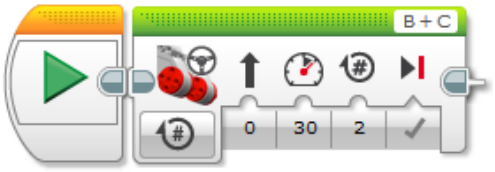
2. The **Block Type** displays an icon that indicates the action component being configured by that block. In this example, a Steering Block (*a.k.a. – Move Steering Block*) is being used picturing two Large Motors and a steering wheel as the icon.
    - Similar to the Steering Block, is the Tank Block. Both provide simultaneous control of two Large Motors, but the approach differs. Our exercises focus on Steering Blocks.
    - In place of the **Direction** field (see bullet #4), a Tank Block has a second **Power** field (see bullet #4) for individual power control of each Large Motor, which for our robot equates to individual forward/backward control of each wheel.
3. Selecting a **Mode** sets the type of **Input Values** that will be available to set parameters for that block. In this example, the default **Mode** of "Number of Rotations" is selected. Notice the number (#) inside the rotation (circular arrow) symbol that makes up the icon.
4. The available **Input Values** will vary according to which **Mode** type has been selected. In this example, the **Mode** is set to "Number of Rotations." This sets the **Input Values** (from left to right) to be **Direction**, **Power**, **Rotations**, and **Brake**. Changing (or leaving) the values in those fields is what determines the parameters of influence that block has. For our robot, it's how the Large Motors are programmed and thus affect how the robot will move according to wheel rotation. More specifically…
    - **Direction**: By entering a plus, or minus value of up to 100 in this field, the robot will either turn left, or right, and the icon (arrow) above it will also change to reflect those values. In this example, with a value of zero (neither plus, or minus), the robot will travel straight forward as indicated by the icon (arrow) above It.
    - **Power**: By entering a plus, or minus value of up to 100 in this field, the robot will either travel forward, or backward, and the icon (speedometer) above it will also change to reflect those values. For our robot, should the value entered be a minus, the robot will move backward, plus the arrow icon above the **Direction** field will change to pointing downward indicating that the robot will move backwards.
    - **Rotations**: By entering a plus, or minus value in this field, the robot will either travel forward, or backward. But for this field, no icons will change to reflect the field entry.
        - Also to note, should the value entered be a minus, the robot will go backward, but the arrow icon above the **Direction** field WILL NOT change to reflect backward travel as it does for a minus value in the **Power** field.
    - **Brake**: This field is strictly a select field. You either select a "Stop" or a "Coast" action for when the previous fields have run their programmed parameters. In this example, the Stop action is selected for a quicker stop, whereas the Coast action allows the Large Motors to continue turning with inertia.

IMPORTANT: A key thing to remember is that the **Mode** and **Input Values** are common to nearly every block (though not necessarily in the same place), and critical to proper EV3 programming. Also, taking note of icons that are above a field, or are part of a selection field (**Mode** as an example), will help you understand each blocks function and configuration properties.

Exercises – Using the Action Block's: [Separate Videos]

NOTE: For all the following exercises where the robots speed (Power) is concerned, for safety's sake, we'll slow things down and operate at a setting of 30, from the default of 50. Depending on the exercise, this setting could be a -30. To run your programs, click the Start button, or the "Download and Run" button on the Hardware Page (ref page 12). I encourage experimenting with configurations as you progress through these exercises.

1. Move Straight [Video 07]
    a. Use a Steering Block to move 2 rotations forward.
        i. Example at right shows the **Mode** set to its default "Number of Rotations," but two of the **Input Values** have changed from their default.
            1. **Power** set to 30 from 50.
            2. **Rotations** set to 2 from 1.

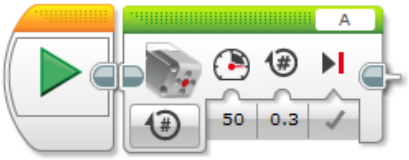


    b. Use a Steering Block to move 2 rotations backward.
        i. Same as above, except change the **Power** setting to -30.
        ii. Note the change in the Speedometer and the Direction icons.
    c. Run your programs and note wheel rotations.

2. Make 90° Turns [Video 08]
    a. Use a Steering Block to make a 90° right turn.
        i. Basically, the same settings as "Move Straight," but with a **Direction** setting change.
            1. **Direction** set to 50 from 0.
            2. Note the icon change.

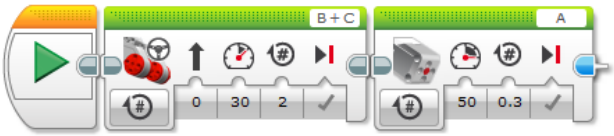
    b. Use a Steering Block to make a 90° left turn.
        i. Basically, the same as above, but with a Direction setting change.
            1. **Direction** set to -50 from 0.
            2. Note the icon change.
    c. Run your programs and note accuracy of 90° and tweak if needed.

3. Move The Arm Up And Down [Video 09]
    a. Use the Medium Motor Block to move the arm up.
        i. Set up: Set arm down in a horizontal position.
        ii. Change only the **Input Value** of number of rotations from 1 to .3 (will show as 0.3).



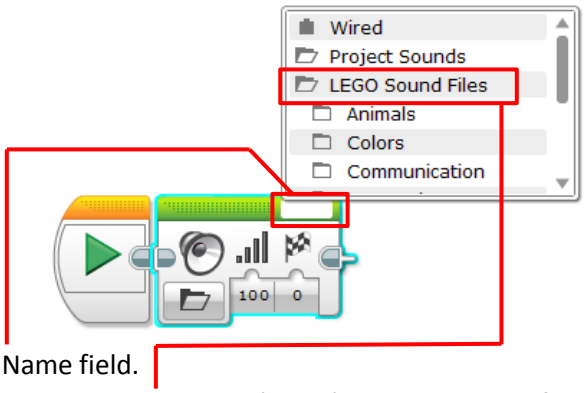
    b. Reset the **Input Value** from 0.3 to -.3 (will show as -0.3).
    c. Run your program and note accuracy of movement and tweak if needed.
    d. What's needed to move the arm up, then down as one program?
        i. Hint: can't do it with just one.

4. Combine a Move and Arm Action [Video 10]
    a. Connect a Steering Block and a Medium Motor Block to move straight forward, then move the arm up.



    b. Using what you've learned, set your Steering Block to move straight at a Power of 30 for 2 rotations and then lift the arm from a horizontal, to a vertical position.
    c. Run your program.
    d. Now add a vertical to horizontal arm move.

5. Play with sounds [Video 11]
    a. Set up: Make sure the sound on your laptop is on and turned up enough to hear.
    b. Using the Sound Block, select a sound to play.
    c. By default, the **Mode** is set to Play File; leave that *and* the **Input Values** as is.



    d. Now select a sound file to play clicking the File Name field.
    e. From the pop-up menu, select anything under the LEGO Sound Files (there's a lot to choose from).
    f. When selected, your sound should play on your computer.
    g. To hear it on your robot, run the program.

NOTE: The Sound Block files (and Display Block files) sometimes don't play on the Brick as they should. Normally, selecting a different sound file, running it (likely no sound still), then going back and running your original selection will usually do the trick. Another way of fixing the issue is also outlined in bullet #6 of the "Downloading to the Brick / Hardware Page Details" section on page 12 and [Video 04].

    h. Have some fun and try out several of the sounds.

IMPORTANT: Because some exercises may require more space, you'll probably want to run your bot unplugged from the computer. Refer to "Running Untethered" on page 14 for details on how to do that.

Orange Flow Block Basics: [Video 12 *includes follow-along programming*]

Note: As you experienced in the "Combine A Move And Arm Action" exercise, when blocks are connected consecutively to form a program string and then executed, the first block in line will run its parameters, then hand off control to the next block, and so on. This characteristic is modified when Flow Blocks are used in a program string.

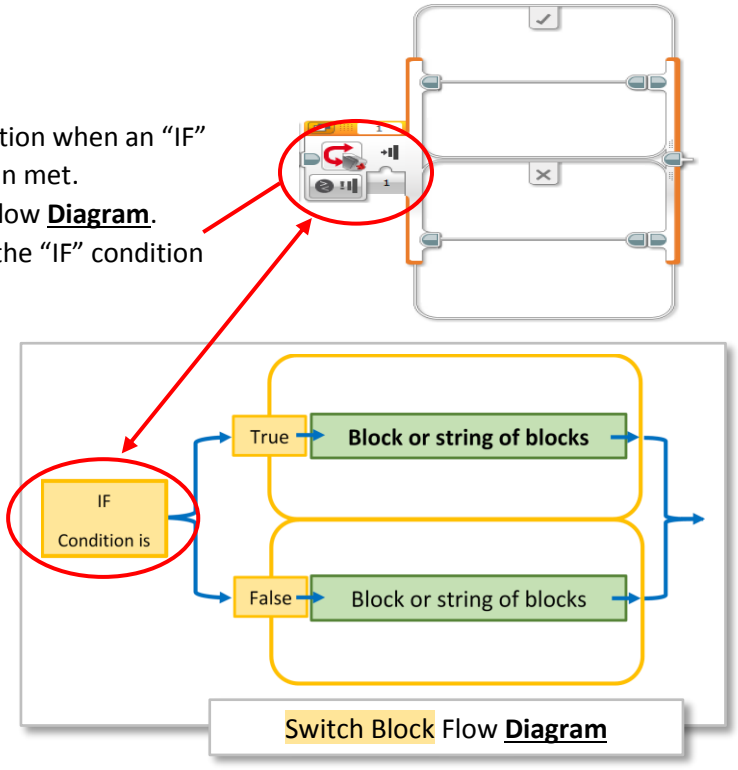
The influence of a Flow Block on a program is entirely dependent on the block, or blocks, being used within them (literally), or connected to them. Basically, without other blocks, Flow Blocks cannot be used. But when used, they offer greater program functionality while having fewer parameters to configure than most other blocks.

There is, however, one particular Flow Block that is used with every program…the **Start Block**. Notice that it has an orange bar on top; this identifies it as a Flow Block the same way a green bar identifies the Action Blocks. You've already used it, and by default, whenever you open a new project or program, it's already placed on the Canvas.

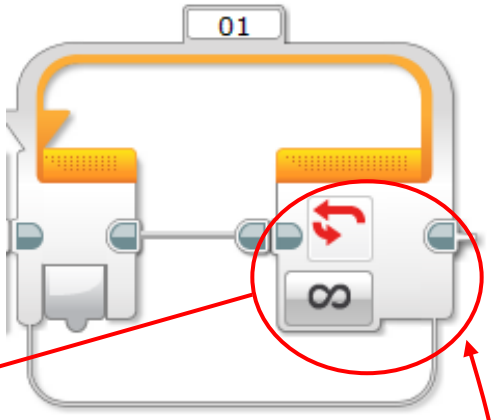
The next Flow Blocks we'll touch on are *each very different* by what they do, but have a common thread of depending on a "Condition" to be met as part of their functional parameters. This "Condition" is set by configuring the **Mode** and **Input Values**. We'll get into more specifics in the following exercises, but for now, here's some brief descriptions.

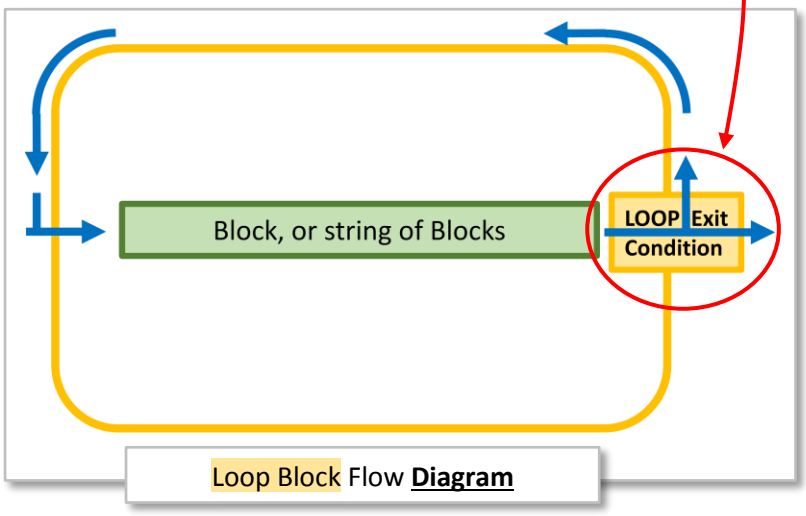
1. **Wait Block**
   a. Specifies a "Run Until" condition for the block that precedes it.
   b. Similar to the Action Block, Flow Blocks have a **Mode** selection field, which affects the **Input** field(s).

2. **Switch Block**
   a. Provides the ability to change a course of action when an "IF" condition (generally a True or False) has been met.
      i. Note blue arrow path in the lower Flow **Diagram**.
   b. The **Mode** and its **Input Values** determines the "IF" condition (compare with diagram).
   c. Blocks must be placed inside the Switch Block to have influence.
   d. Though the diagram suggests Green Action Blocks are to be used, Flows Blocks are also applicable.



3. **Loop Block**
    a. Provides continuous operation of a block, or string of connected blocks, by looping back and starting over.
        i. Note blue arrow path in lower Flow **Diagram**
    b. A "Loop Exit Condition" must be set to end the looping process, and that condition must be met before handing off control to the next block in line.
    c. The default "Loop Exit Condition" is "Unlimited" ∞ which will loop a program continuously until either manually stopped, or the batteries run out.
    d. The **Mode** and its **Input Values** determines the "Loop Exit Condition" (compare with diagram).
    e. Blocks must be placed inside the Loop Block to have influence.
    f. Though the diagram suggests Green Action Blocks are to be used, Flows Blocks are also applicable.



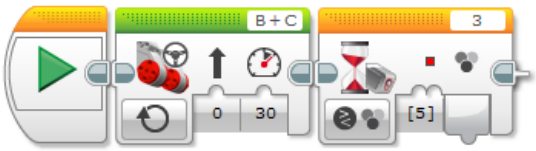


Exercises – Using the Flow Blocks: [Separate Videos]

1. Blink the lights [Video 13]
    a. Connect a Brick Status Light Block, and a Wait Block to set the lights on the Brick to blink orange "Until" 5 seconds have passed.
    b. Example at right shows the Brick Status Light Block in its default configuration, and the Wait Blocks **Input Value** set to 5 seconds. To set the **Input Value**…
        i. Highlight the number 1 in the **Input Value** field and type in 5.
        ii. Note the clock icon in the **Mode** field and the hour glass icon above it. This indicates the Wait Block is configured to "Run Until" a value of time (seconds) has passed,
        iii. Also, the Brick Status Light Block by itself runs so quick, that it appears nothing happens; this is why a Wait Block must be used to create a run time duration for the lights to be seen.
    c. Run your program and watch the green display light change to orange and blink for 5 seconds.

IMPORTANT: Because some exercises may require more space, you'll probably want to run your bot unplugged from the computer. Refer to "Running Untethered" on page 14 for details on how to do that.

2. Color Sensor Stop [Video 14]
    a. Set up: Place a 3" strip of color tape across the path of a straight movement.
    b. Connect a Steering Block and a Wait Block and configure to move straight "Until" the Color Sensor detects the tape color you're using.
    c. Example at right shows the Wait Block **Mode** configured to "Compare Color" with the default **Input Value** color as red. To configure this…

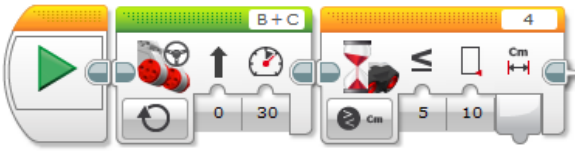


        i. Click **Mode** and from the drop down menu select *Color Sensor> Compare> Color.*
        ii. Click the **Input Value** and select the color of your tape if it's other than red.
            1. Uncheck the red if not needed.
    d. It also shows the Steering Block **Mode** set to ON.
        i. Using what you've learned, reset the **Mode** to ON, and the Power Input Value to 30.

IMPORTANT: The **Mode** of "ON" only works when there is a follow-up block that effectively sets the run duration. Otherwise, by itself, it has no idea how long to run, so it would do nothing.

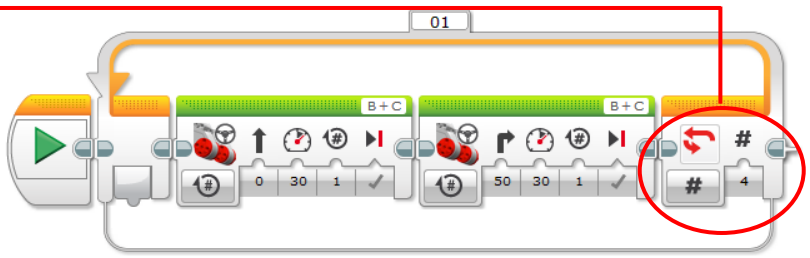
    e. Run the program and watch your robot stop after detecting the color strip.
    f. Bonus: Add a Steering Block to the end and set the **Mode** to "Off."
        i. Run the program and note the difference.
        ii. Now set *that* Steering Blocks Brake **Input Value** to Coast (X)…run and note the difference.
    g. Bonus2: Delete the Wait Block and the second Steering Block, then run program.
        i. What happened and why? (refer to the "IMPORTANT" message above).

3. Ultrasonic Sensor Stop [Video 15]
    a. Set up: Place the Color Block in the path of a straight movement.
    b. Connect a Steering Block and a Wait Block to move straight "Until" the Ultrasonic Sensor detects the Lego Color Block, or other obstacle.
    c. Example at right shows the Wait Block **Mode.** configured to "Compare Distance" in centimeters with the **Input Values** set to "Equal To, or Less Than" (**≤**) a distance of 10 centimeters. To configure this…



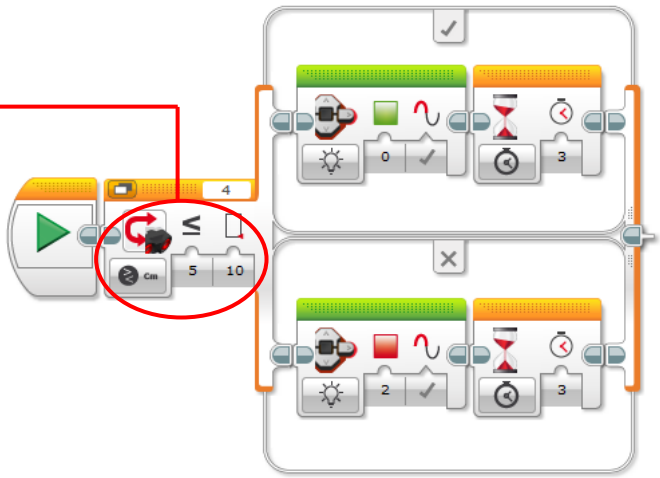
        i. Click your **Mode** and from the drop down menu select *Ultrasonic Sensor>Compare>Distance Centimeters.*
        ii. In the first **Input Value**, select the **≤** symbol (#5 selection) from the drop down menu.
        iii. In the second **Input Value**, click in the field to highlight, then type in the number 10.
    d. As with *Part 1*, reset your Steering Block **Mode** to ON.
    e. Bonus: Add a Steering Block to the end and set the **Mode** to "Off."
        i. Run the program and note the difference.
        ii. Now set *that* Steering Blocks Brake **Input Value** to Coast (X)…run and note the difference.

4. Move in a square [Video 16]
    a. A square = a straight move + a 90° turn x 4.
        i. You could do this with eight blocks in a string, or…
    b. Place 2 Steering Blocks inside a Loop Block.
        i. Configure the first Steering Block to move straight for 1 rotation.
        ii. Configure the second Steering Block to turn 90°.
    c. Now set the Loop Exit Condition (**Mode** and **Input Value**) to run the loop 4 times…
        i. Click **Mode** and select "# Count" from the drop down.
        ii. Highlight the **Input Value** and type 4.



    d. Run the program and watch your robot move in a square(ish) way.
        i. Each robot, as well as environmental conditions, will influence accurate operation. So you may need to dial in the 90° turn to be an actual 90° turn.
        ii. Experiment with the **Input Values** for a better 90° turn.

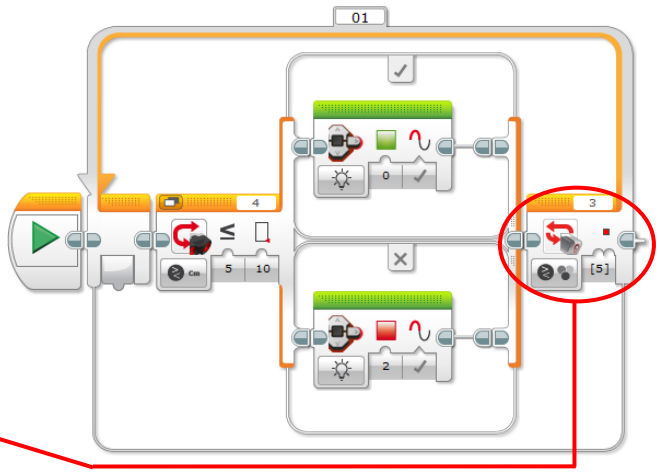
5. Change the blinking lights – *Part 1* [Video 17]
    a. Using a Switch Block, set the Switch "IF." Condition, (**Mode** and **Input Values**) to Ultrasonic Sensor.
        i. Click **Mode** and select from the drop down *Ultrasonic Sensor> Compare>Distance Centimeters.*
        ii. Change the first **Input Value** to 5 (≤).
        iii. Change the second **Input Value** to 10.



    b. Place a Brick Status Light Block in the upper, and lower area of the Switch Block.
        i. Change the first **Input Value** to green for the top block; and red for the bottom block.
    c. Add Wait Blocks to the end of each Brick Status Light Block and change the Input Values to 3.
    d. Alternate running the program with, and without an obstacle (like your hand) within 10 centimeters of the Ultrasonic Sensor to see the colors change.
        i. Note that the program will only run once…why?

Think about this; even though the Switch Block has multiple blocks within it, it's acting as if it were one block. So when it has run its parameters and acted on either the IF True, or IF False condition, it's completed what it's been programmed to do and tries handing off control to the next block if there had been one.

6. Change the blinking lights – *Part 2* [Video 18]
    a. Set up: Fold in half a roughly 6" piece of color tape so that you can handle it without being stuck to it.
    b. Place a Switch Block inside a Loop Block .
        i. Note the Loop Block expands to accommodate the Switch Block.
    c. For the Switch Block, use the exact same configuration as in the previous exercise minus the Wait Blocks.
    d. For the Loop Block, set the Loop Exit Condition **Mode** to *Color Sensor>Color*, and the color **Input Value** to whatever your tape color is.



    e. Run the program and alternate an obstacle (like your hand) back and forth in front of the Ultrasonic Sensor and watch the Brick lights change accordingly.
        i. Do the lights blink…yes or no? Why?
    f. To end the program (exit the loop), place the color tape strip under the Color Sensor.

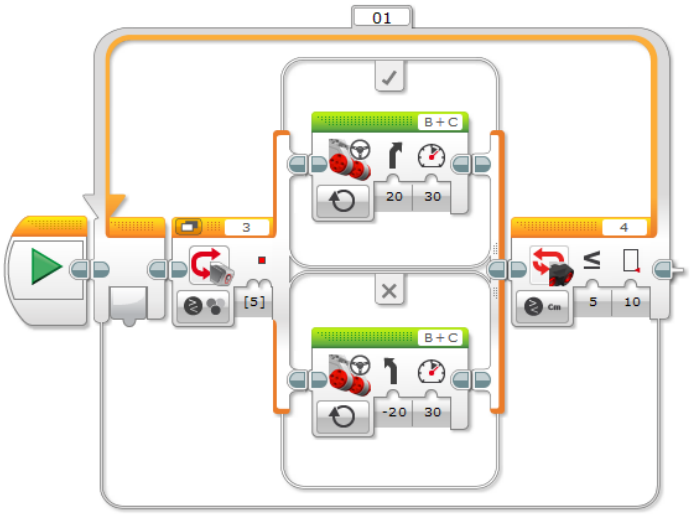
Think about this: Even though the Brick Status Light Blocks are set to "Blink," notice that they DON'T blink. Because the Switch Block is now in a Loop Block, and the Brick Status Light Blocks have not been given a time duration to blink via a Wait Block, the Ultrasonic Sensor is searching so fast and frequently, that the lights simply don't have time to blink. They keep receiving information from the Ultrasonic Sensor so quickly that they can't complete the act of a blink. Try adding a Wait Block set for 3 seconds after each Brick Status Light Block, run the program, alternate an obstacle, and notice that the lights WILL blink for the duration set by the Wait Block, but the Ultrasonic Sensor will not search until they have completed their blinking.

For the next exercise, (Line Follow) some understanding of what's really happening is in order. First, it's going to be more of a challenge rather than an exercise, in that I'm NOT going to have reminders of how to set configurations. I'll only point out what needs setting. With that said…

Though what we are attempting is known as a "Line Follow," the robot isn't actually following the line itself; it's technically following the edge of the line by reacting to color changes. The principle is that when a particular color is "seen" or "not seen" by the Color Sensor, the robot is to veer either right or left accordingly. So when properly configured, it waddles down the line as it "sees" the color and veers away from it; then when it "doesn't see" the color, it veers back toward it. This action is repeated continuously and thus, you get a waddling robot.

Whether it's configured to veer left or right for each instance will determine if it follows the left or right edge of the line. Per the program description and screen shot below, we will configure it to follow the right edge of the line. Note the configuration and decipher what makes it a right edge line following program? Once you've dialed in your configurations for smooth operation (below example is a starting point), flip your Steering Block settings and watch how it follows the left edge of the line.

7. Follow the Line [Video 19]
    a. Set up: Lay a strip of color tape at least 2 feet long (double if you have the space, and it doesn't have to be perfectly straight) and place the Lego Color Block at the end directly in the way of the Ultrasonic Sensor on the left side of the line.
        i. Note that with the Color Sensor positioned on the right side of the robot, the robot itself will be positioned to the left of the line as it waddles.



    b. Place a Switch Block inside a Loop Block.
    c. Configure your switch condition to the Color Sensor and the color of your tape line.
    d. Place Steering Blocks in the top and bottom of the Switch Block.
    e. Configure your Steering Blocks to "ON," and to veer right for the top, and veer left for the bottom.
        i. NOTE: This configuration is a rough start, you should tweak your settings (trial and error) for optimum performance.
    f. On the Loop Block, configure the Loop Exit Condition to Ultrasonic Sensor with **Input Values** of "less or equal to" and 10 centimeters.

Final Challenge:

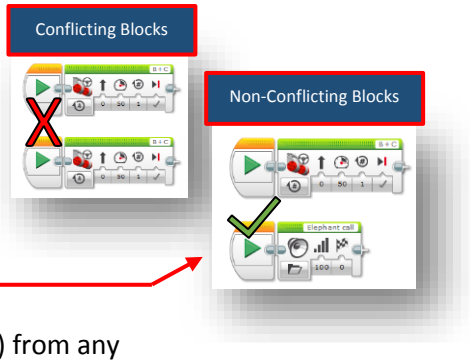
OK, you've learned what the Green Action and the Orange Flow Blocks do, how they are the same and yet different, and how they all work in concert. So now we're going to put what you've learned to some practical use. Remember back in the "Programming Workflow" section (page 9) where the "Basic Programming Flow Model" was described and the "Pseudo Programming" was outlined? We're going to revisit that in the form of a challenge.

Below is a pseudo program sequence for you to decipher and build your own program solution. Remember to apply the Programming Flow Model as you progress. Everything you need to know has been covered in the previous instructions and exercises, though you'll need to tweak some of the settings we used to achieve a few of the goals. Here's one hint just for kicks…less of one thing and more of another will make the sharp 90° turns used in our squares into arcs. With that said, here's the pseudo programming for you to decipher -

1. Starting with the arm raised, follow a line (at least 2') to the color cube and stop
2. Lower arm to capture the cube
3. Reverse in an arc and stop
4. Raise the arm back up and make a victory sound

When you feel you've conquered the challenge, watch [Video 20] and see that there's not always just one answer. Because it's very likely that my answer will be a bit different than yours.

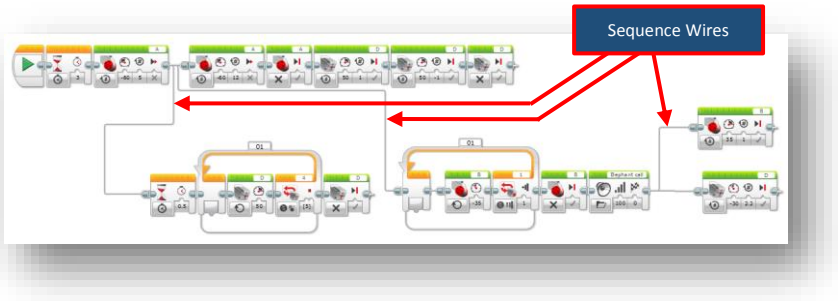
More Fun Stuff – Simultaneous Programs: [Video 21]



By adding a second Start Block to the canvas, you can create a second (or more) program string. **WARNING**: *Each string MUST have different block types running to prevent conflicting commands.* For instance…

- A Steering Block used at the same time will cause conflicting commands and not work.
- A Steering Block in one string and a Sound Block used in the other will work, since that won't cause conflicting commands.

Using **Sequence Wires** allows you to start a second (or more) program string(s) from any point of a program.



For instance, you could have your robot moving and then at some point have your robot make animal noises without stopping.

- You'll still need to watch for conflicting blocks!
- Watch at the end of [Video 21] "Elefonte On The Move" for a demonstration and more information, which includes an example of what can be built by adding pieces from a LEGO Mindstorm Expansion Set.
- Screen shot is of actual "Elefonte On The Move" program using **Sequence Wires** to accomplish simultaneous robotic actions.

Brick Maintenance: [Video 22]

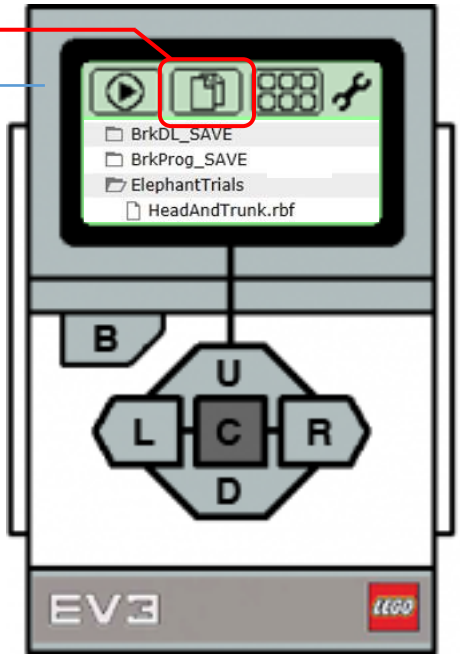
When we're all done with our sessions and lots and lots of programs have been run on your robot, it's time to clear all that out to make way for the next session of programming fun.

There are two ways in which you can delete post-session programming from the Brick; either from within the EV3 Software, or from the Brick itself. Both are quick and worth learning.

Clearing Post-Session Projects From The Brick, using the Brick:

IMPORTANT: **PLEASE DO NOT DELETE** the folders **BrkDL_SAVE**, or **BrkProg_SAVE** that list under the File Navigation Tab. These are default folders.

1. With the Brick on and depending on your starting point, navigate [L]eft or [R]ight to the File Navigation tab.
2. Navigate [D]own to highlight the Project that needs to be deleted.
3. Once highlighted, press the [C]enter button until the Trash icon appears.
4. Press the [C]enter button again to get the "are you sure?" window.
5. Press the [R]ight button to highlight the check mark.
6. Press the [C]enter button one last time to delete selected Project.



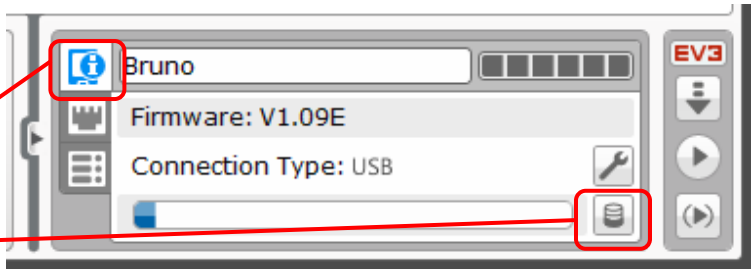
After completing the process, turn off the Brick

1. Press the [B]ack Button until the Shut Off screen appears.
2. Tab [R]ight to highlight the check mark.
3. Press the [C]enter Button to initiate shut down.
4. The light will turn red, then off when shut down.

Clearing Post-Session Projects from The Brick, using the EV3 Software:

Note: This method can be done consecutively without closing the Memory Browser

1. With the EV3 software launched, open a new Project
2. With the Brick turned on, plug it in to the computer via the USB Cable
3. Locate the Hardware Page In the lower right hand area of the EV3 Programming Page (screen shot)
4. Click the Brick Information Icon (may already be selected)
5. then click the Open Memory Browser icon



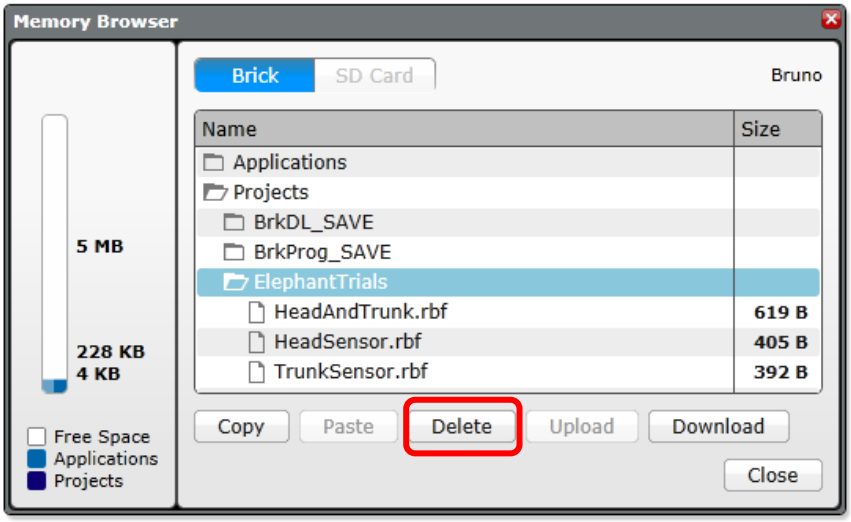
The **Memory Browser** window will open and if there is no SD Card involved, the Brick tab (next screen shot) will be selected (highlighted blue) by default.

IMPORTANT: **PLEASE DO NOT DELETE** the **Applications** folder, or the folders **BrkDL_SAVE**, or **BrkProg_SAVE** that list under the **Projects** folder. These are default folders.

NOTE: All other folders not mentioned above have been created during a session and should be deleted. In the example screen shot below, **ElephantTrials** is a saved project and the listings within that are titled programs. Quite often, projects will not be saved and will show up as **Project**, **Project2**, **Project3**, and so on. And if the subsequent programs are not titled, they will show up in the same manner, **Program**, **Program2**, **Program3**, and so on. Deleting the Project, will also delete the affiliated programs, but only one project may be deleted at a time.

To delete a Project

1. Highlight project by selecting it ("ElephantTrials" for this example).
2. Click the **Delete** button that becomes active after selecting.
3. Repeat steps 1 & 2 for multiple Projects.
4. Unplug USB from Brick.
5. See below for Clearing several Bricks quickly using this method.



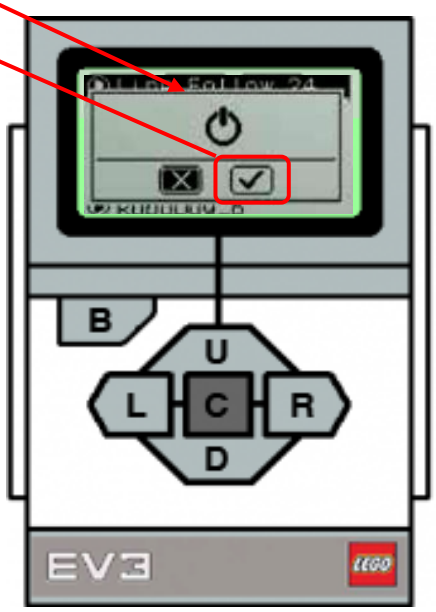
Clearing several Bricks quickly:

In the case where a whole class has been doing Mindstorm Robotics, each Brick will need to be cleared of its programs before the next session. You could either clear the programs using the Brick method, which is fairly quick, or you could use the software method, which is slightly quicker. The thing to remember is that once you're set up, DO NOT close the Memory Browser while unplugging and plugging Bricks in. This is where you gain the speed.

1. Have all your Bricks nearby and turned on
2. Once one Brick has been cleared using the software "To delete a Project" steps above, unplug the USB from the Brick (not the computer), and plug the next one in
3. Repeat step 2 until all Bricks have been cleared.

After clearing Projects and Programs from a Brick, turn off the Brick

1. Press the [B]ack Button until the Shut Off screen appears
2. Tab [R]ight to highlight the check mark
3. Press the [C]enter Button to initiate shut down
4. The light will turn red, then off when shut down



Added Resources:

Curriculum:

Carnegie Mellon's Robotics Academy – Introduction to Programming EV3 Teacher's Guide

- http://www.education.rec.ri.cmu.edu/content/lego/ev3/files/EV3%20teachers%20guideWEB.pdf

Online Training:

Carnegie Mellon Robot Academy

- http://www.education.rec.ri.cmu.edu/content/lego/ev3/
- http://education.rec.ri.cmu.edu/previews/ev3_products/ev3_curriculum/
    - An introduction to Programming LEGO® MINDSTORMS® EV3 (a.k.a. EV3 Trainer)
    - A series of interactive training modules consisting of short videos, assigned tasks, and multiple choice follow-up questions. Presented within context of real world robotics, same materials used in 6-week online training course, but without advanced challenges or instructor-led class time, aimed at educators. [engaging]

DrGraeme.org – "Fun with LEGO EV3 for Absolute Beginners" video tutorials

- https://www.youtube.com/playlist?list=PLFd4sjeVCkV_sN15Y4Jiaz4v3Csk9cd3W
    - A series of 47 video tutorials, ranging from under a minute to over 22 minutes in length, for over three hours of inductive training. Aimed at Australian middle school students, robots used in these examples are less complex and functional than the standard educational model. Self-described as a very gentle introduction to building and programming LEGO EV3 Mindstorms Robots [maybe a little too gentle].

Educational Robots for Absolute Beginners – EV3 Edition / Rowan University

- https://cs4hsev3robots.appspot.com/
    - Online workshop materials consisting of self-paced courses specifically designed with K-12 teachers as a 5-week long workshop. Includes short instruction videos, self-test questions, and assigned exercises with an estimated time commitment of 4-6 hours per course week. Workshop built using the Google Course-Builder Tool and therefore requires a free Google account to register for access. Practical step-by-step instructor-led video presentations [thorough introduction].

EV3Lessons.com – by Droids Robotics

- http://ev3lessons.com/lessons.html#en-us

STEMcentric – EV3 Tutorial

- http://www.stemcentric.com/ev3-tutorial/
    - Ninety minutes worth of video tutorials varying in length from 2-18 minutes, with some assigned tasks but no substantive follow up/review. Less structured than Rowan University and Carnegie Mellon online training resources.

STEM Robotics 101 EV3-Portland State University [free registration required]

- http://stemrobotics.cs.pdx.edu/node/2643?root=2643