ADVANCED ROBOT PROGRAMMING LEGO® MINDSTORMS – EV3

Presented by:

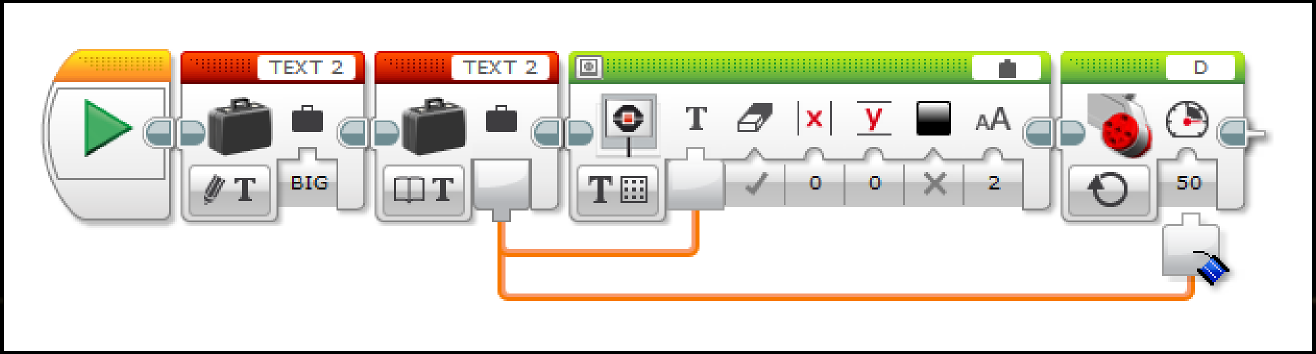
Tom Bickford
Maine Robotics

Friday, 10/16/2015 1:30-3:30

- Description:
  - So you've been using robotics in your classroom but want to take it farther? This workshop will use the LEGO MindStorms kits to introduce variables, conditional pathways, data display, timers, mathematical calculations, and data logging.

DATA WIRES

- DATA WIRES allow you to communicate information from one block to another within a program. They can transmit TEXT, NUMBERS, ARRAYS, or LOGIC (true/false) between components.
- If you try to connect incorrect data types, you will not be able to make the connection
- Here you see a DATA WIRE going from a TEXT VARIABLE to a TEXT DISPLAY, but if you try to connect it as an input for MOTOR SPEED, it won't allow the connection
- Notice also that it allows you to take the same value more than one time out of the same source



VARIABLES

- Variables are place holders in a computer program
- They can be used across other structures within the program, such as loops and conditional statements. They can also be used in different segments of the same program
- Variables are defined, read, and written to all using the same variable block

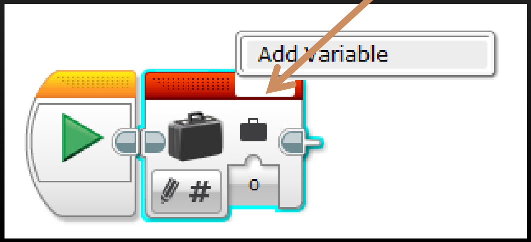
- They are not used to store information between programs or saved for later use. Once the program is ended, the variable information is lost

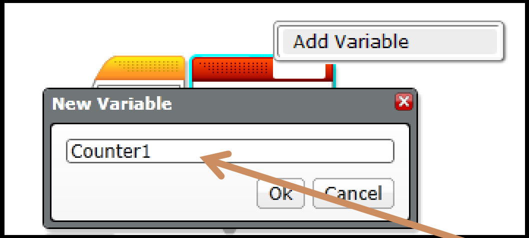
- Variable blocks are found in the red DATA options section of the program

DEFINING A VARIABLE

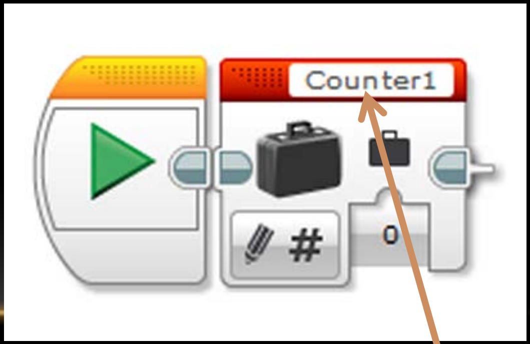
- Click on the empty field at the top of the block
- Any already defined variable names will be shown, or you can enter the name of a new variable
- The type of variable is defined the first time that variable is used, then it will remain the same for the rest of the program
  - Name the new variable “COUNTER1” then WRITE the value 1 to it, it is now a NUMERIC variable and will remain so for the rest of the program.
- Once you define “COUNTER1” as a numeric variable, if you try to write a text value to it later, it will give you an error message.

Click here





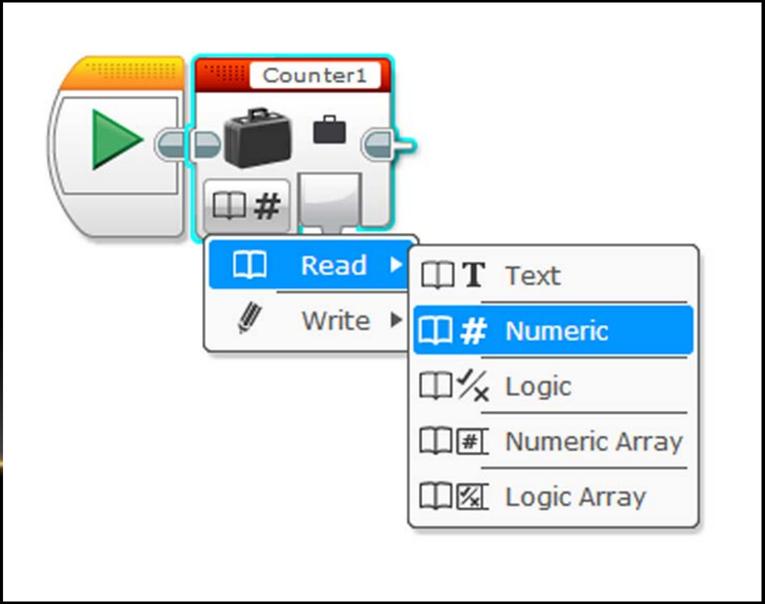
Name the variable here



Name now shows

TYPES OF VARIABLES

- In a text variable you are saving or using what is called a STRING, a series of characters
- In a numeric variable, you will be saving a NUMBER, including decimal and negative numbers
- In a logic variable, you will be storing a BINARY STATE (true/false)
- In Numeric and logic arrays, you can store those two types of data, but can have many pieces of data associated with a single variable.

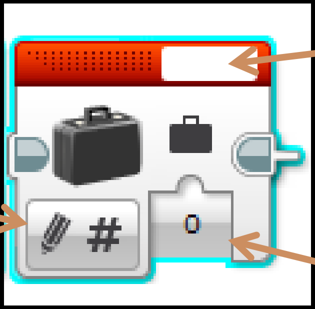


WRITING TO VARIABLES

Select WRITE and also select the TYPE of variable

- Text
- Numeric
- Logic (yes/no, 1/0, true/false)
- Numeric array
- Logic array

The WRITE block



Name the variable here, or select from already named variables

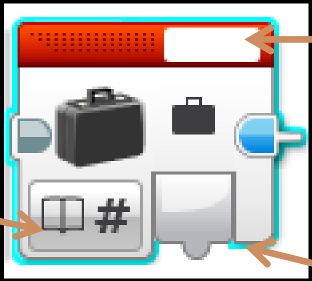
Enter value or connect a data wire here to give the variable a value

READING FROM VARIABLES

Select READ
and also select the TYPE of variable

- Text
- Numeric
- Logic (yes/no, 1/0, true/false)
- Numeric array
- Logic array

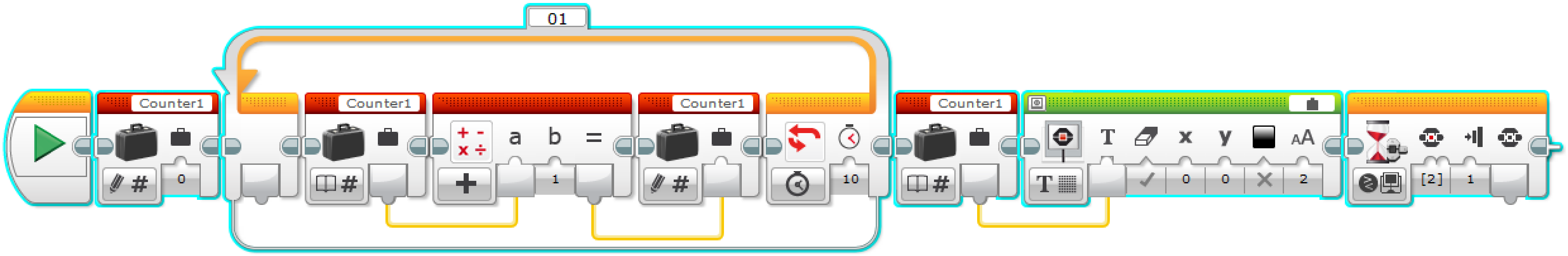
The READ block



Name the variable here, or select from already named variables

Take the value of the variable here, and use it as an input for another block

CONNECTING VARIABLES INTO A PROGRAM



There are 4 variable blocks used in this program. The first one, outside the loop, is used to initialize or define the variable as "Counter1" and write it with the numeric value of "0".

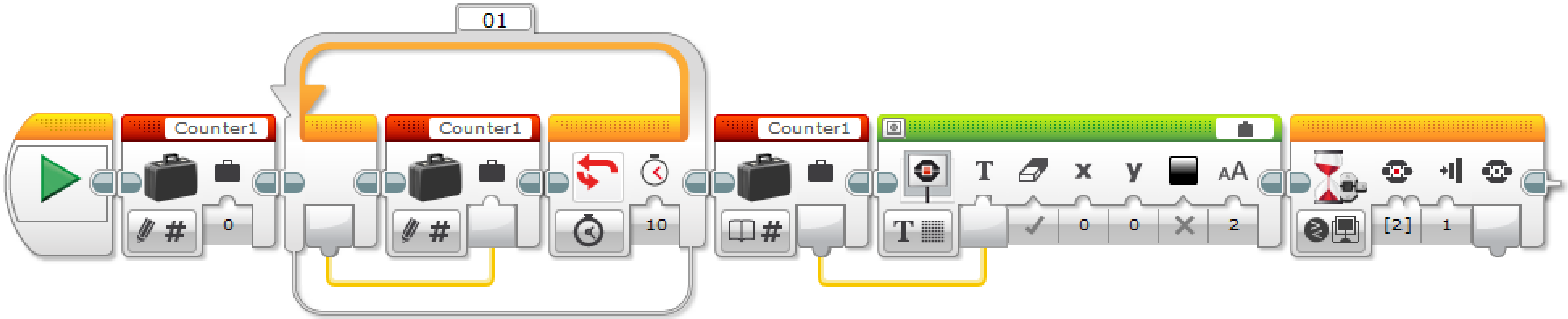
Inside the loop we take the value of Counter1, add 1 to it, and write it back in to the variable. This is a simple counter system for the program (and there are other ways of doing this as well). Each time the loop cycles, it adds one to the variable.

The loop is set to 10 seconds and then exists.

Once outside the loop, we read the value of Counter1 and display it on the screen until a button on the EV3 is pushed.

The LOOP block has a counter built into it, but you cannot take the information out of the loop without using a variable. You could use a single WRITE variable block and keep feeding the loop counter into it, until the LOOP exits.

CONNECTING VARIABLES INTO A PROGRAM



There are 3 variable blocks used in this program. The first one, outside the loop, is used to initialize or define the variable as "Counter1" and write it with the numeric value of "0".

Inside the loop we take the value of the built in LOOP counter and write it to Counter1. Each time the loop cycles, it updates the variable to the counter #.

The loop is set to 10 seconds and then exists.

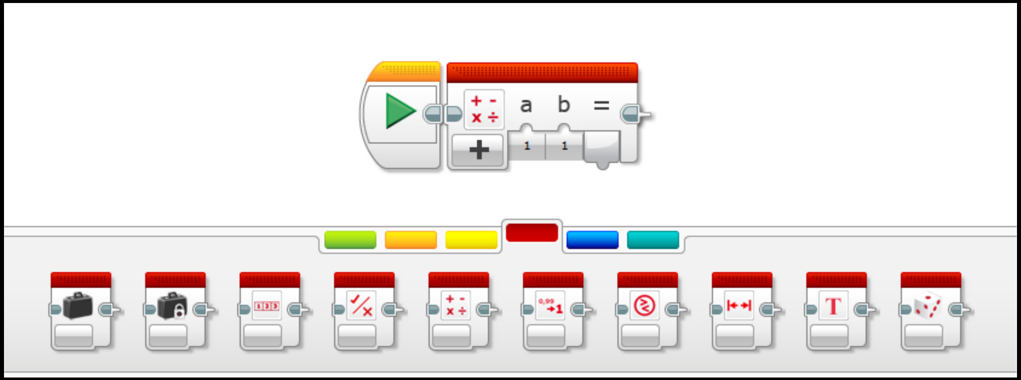
Once outside the loop, we read the value of Counter1 and display it on the screen until a button on the EV3 is pushed.

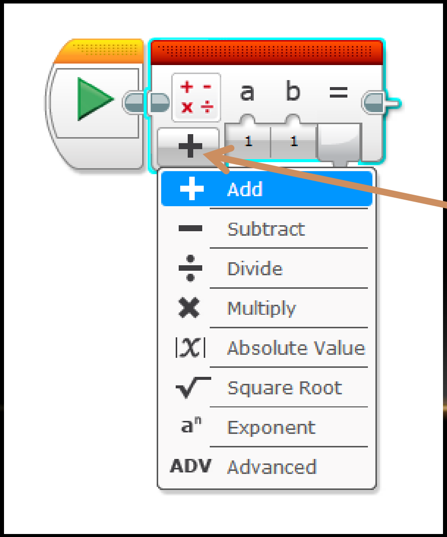
ACTIVITY - VARIABLES

- Build a program that:
    - Uses a variable
    - Counts the number of loops in X number of seconds (you pick X)
    - Displays the number after the loop finishes
    - Try it with the “read, add 1, write” and with the “read off the loop counter”
- For extra credit, add a math block so it displays the number of loops in 1 second.
- Which was faster? The “read, add 1, write” option or the “read off the loop counter”?

MATH BLOCKS

- The MATH block can be found in the red DATA OPERATIONS tab of commands





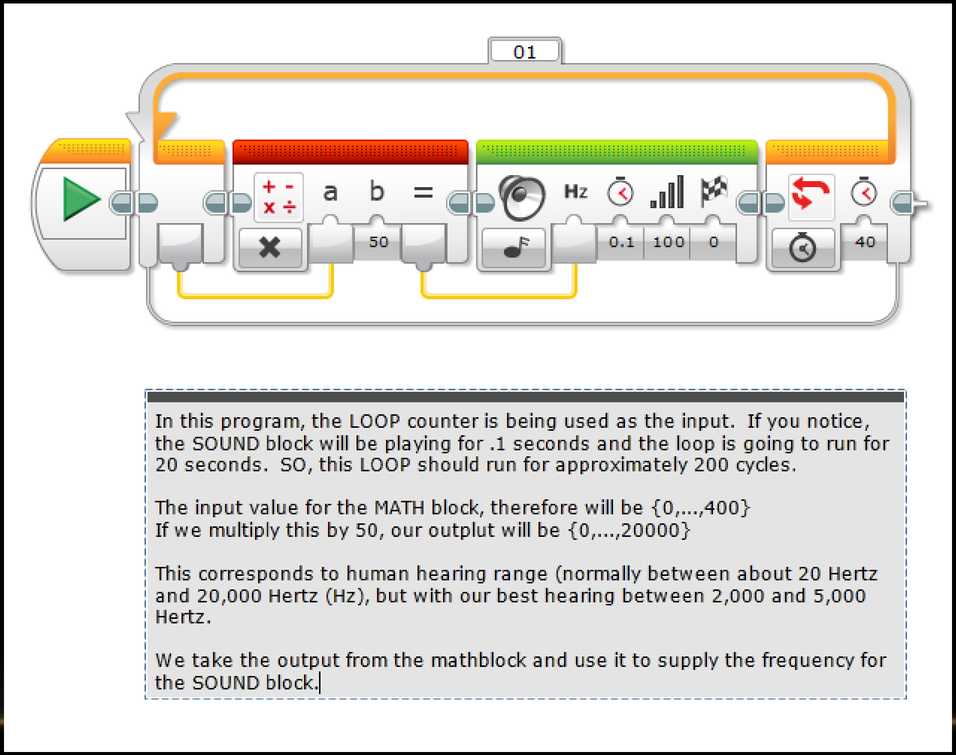
Select which operation you wish to use by clicking on the "+" sign, then selecting your desired option

MATH BLOCKS, CONTINUED

- The ADD, SUBTRACT, MULTIPLY, DIVIDE and EXPONENT options all have <u>two options</u>, one for each number in the equation
  - You can either enter the numbers you want manually (type it in)
  - Or use DATA WIRES to bring the information in (or do one of each)
- The ABSOLUTE VALUE and SQUARE options all have just <u>one option</u>.
- So you can do:
  - 2 + 3
  - 5 – 7
  - 5^3
  - |-6,325|
- In each MATH block, there is an output to connect a DATA WIRE to that will allow you to use the outputted information elsewhere in your program. You could use this to:
  - Control motor speed
  - Control motor direction
  - Change sound output
  - Display the values or change the display

USING THE MATH BLOCK

- Here we see the MATH block multiplying the LOOP counter and feeding that number into the SOUND block in order to give us a sliding tone generator.



In this program, the LOOP counter is being used as the input. If you notice, the SOUND block will be playing for .1 seconds and the loop is going to run for 20 seconds. SO, this LOOP should run for approximately 200 cycles.

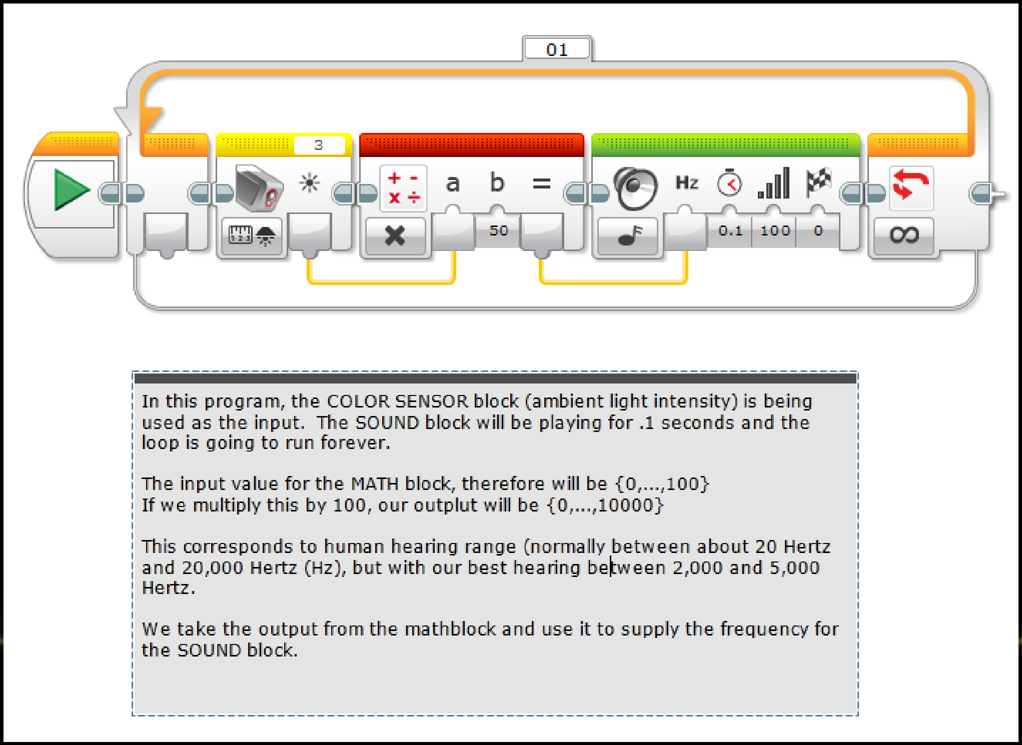
The input value for the MATH block, therefore will be {0,...,400}
If we multiply this by 50, our outplut will be {0,...,20000}

This corresponds to human hearing range (normally between about 20 Hertz and 20,000 Hertz (Hz), but with our best hearing between 2,000 and 5,000 Hertz.

We take the output from the mathblock and use it to supply the frequency for the SOUND block.

MATH BLOCK AND SENSOR VALUES

- Here we see the MATH block multiplying the COLOR SENSOR ambient light intensity and feeding that number into the SOUND block in order to give us a sliding tone generator, based on the light level.



In this program, the COLOR SENSOR block (ambient light intensity) is being used as the input. The SOUND block will be playing for .1 seconds and the loop is going to run forever.

The input value for the MATH block, therefore will be {0,...,100}
If we multiply this by 100, our outplut will be {0,...,10000}

This corresponds to human hearing range (normally between about 20 Hertz and 20,000 Hertz (Hz), but with our best hearing between 2,000 and 5,000 Hertz.

We take the output from the mathblock and use it to supply the frequency for the SOUND block.

ACTIVITY – MATH BLOCKS

- Build a program that:
  - Use a SENSOR value (Light or Ultrasonic sensor works best)
  - Adds, multiplies, subtracts or divides from the sensor value
  - Displays the number on the screen
    - Either as a number or as an object (circle or rectangle) with the value forming either the size or the radius of the object.
  - Put it all in a LOOP

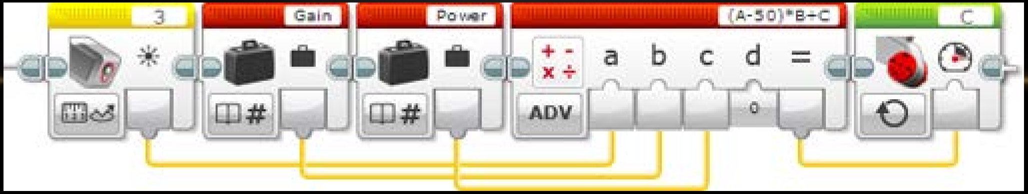
THE MATH BLOCK, ADVANCED OPTION

- In the ADVANCED mode, the MATH block can calculate a mathematical expression using up to four inputs and several math operations in one step.

  Use DATA WIRES to connect up to four Numeric values to the A, B, C, and D inputs. Unneeded inputs can be left blank or 0.

  Click the Block Text Field at the top of the block to enter the mathematical expression in text form to calculate. The expression can include the inputs by name as “A”, “B”, “C”, and “D”, numeric constants such as “50”, and math symbols such as “+”. You can also use functions from the list displayed and additional parenthesis to change the order of operations.

  The result of the expression calculation is output in the result.
- *Example*
- In this program the Math block calculates a motor power using inputs from the Color Sensor and two Variables. The Reflected Light Intensity from the Color Sensor is wired to the A input, and the variables named “Gain” and “Power” are used for B and C. The expression “(A-50)*B+C” in the Math block subtracts 50 from the light intensity, multiplies the result by value of “Gain”, and then adds the value of “Power”.



ADVANCED FORMULA OPTIONS

- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- MODULO
- EXPONENT
- NEGATE
- FLOOR
- CEIL
- ROUND
- ABSOLUTE
- LOG
- LN
- SIN
- COS
- TAN
- ASIN
- ACOS
- ATAN
- SQUARE ROOT

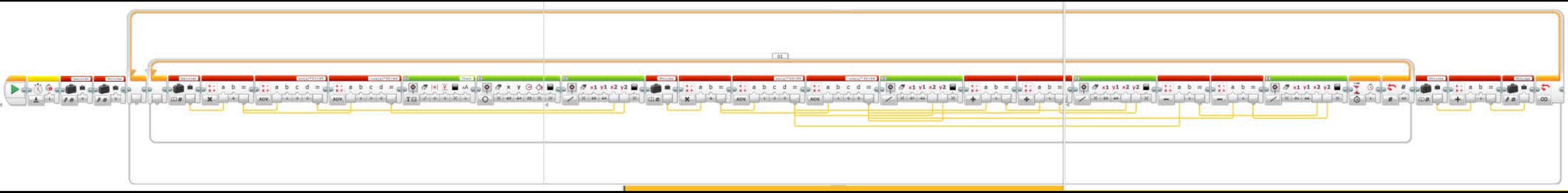
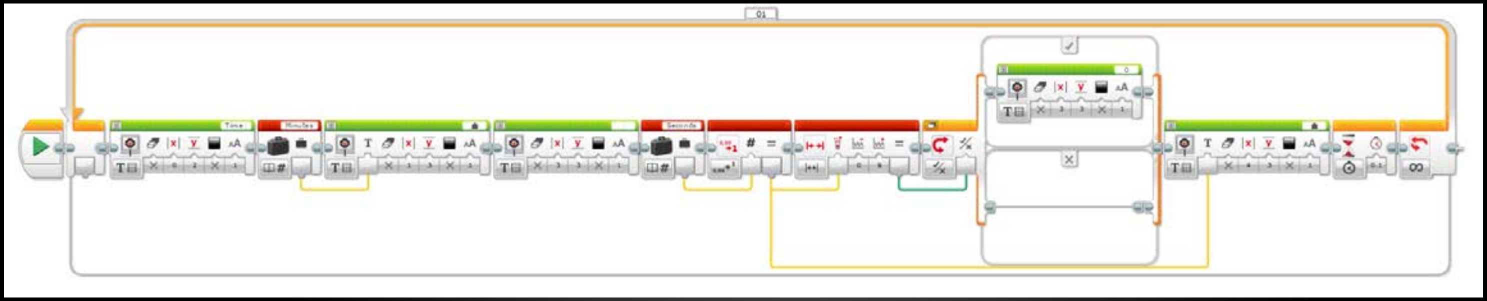
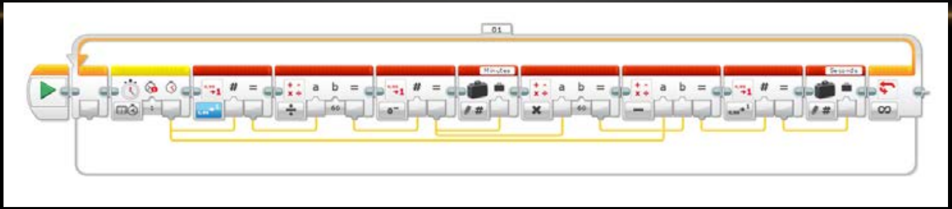
As long as you keep the number of numbers to the A, B, C, and D then you are able to do a lot with this block.

Remember, you don't have to use all 4 numbers. If you only wanted the SIN of a single number, you would still need to use the advanced MATH block option, just leave the other numbers blank

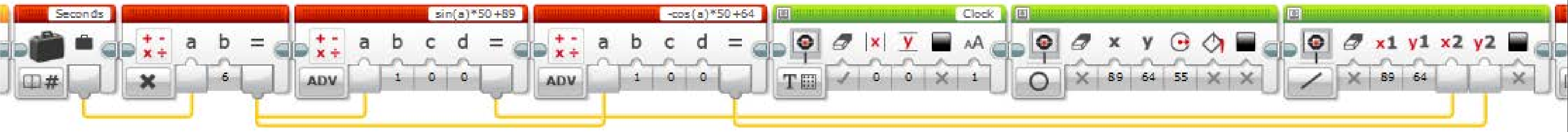
MAKING A CLOCK WITH THE EV3

Here we have a rather large program to make a handed clock using the EV3 and the display blocks.
We have two variables – MINUTES and SECONDS, both of which are numeric variables
We use the ADDITION, SUBTRACTION, DIVISION, MULTIPLICATION, ROUNDING and ADVANCED features of the MATH block.
We've used three different program segments:

1. The main segment that draws the clock face and moves the hands
2. A section that uses a TIMER to set the values of the SECONDS and MINUTES
3. A section that displays a digital readout of the number of minutes and seconds

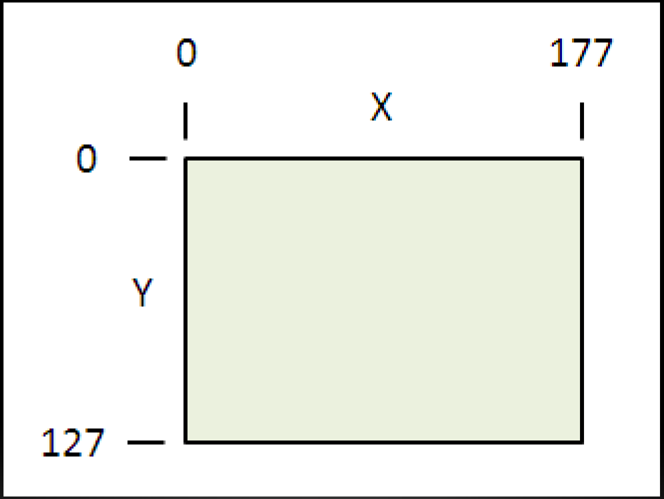
DISPLAYING A CLOCK HAND



- Here we take the value of the SECONDS variable (and it is a decimal number, taken from another section of the program)
- We then MULTIPLY it by 6, so SECONDS {0..60} becomes degrees {0..360}
- We then use the ADVANCED MATH block and the SIN and COS functions to calculate the x and y coordinates for the end of the second hand. NOTE: we have multiplied the SIN and COS output by 50 so we go from having {-1..1} set of numbers to a {-50..50} set of numbers. We also Add 89 to the x value to center on the display and 64 to the y for the same reason.
- We DISPLAY “Clock” in the upper corner
- We DISPLAY a circle on the screen to frame the clock
- We draw a line from the center (89,64) to the coordinates from the ADVANCED MATH block.

THE EV3 DISPLAY FACE

- Within the EV3 Display there are two ways to display. When graphic POINTS, LINES, CIRCLES, or RECTANGLES you use the pixel option
- When displaying text you can use either the pixel or the grid option. The grid option uses lines and character spacing, not pixels. So think of 4th row down, 8th character over. But the counting still starts from the upper left corner.



ARRAYS

- NUMERIC ARRAY

  The Numeric Array type represents a list of numbers. The list has a certain length, and each element in the list is a Numeric value. An array can have any number of elements (limited by the available memory on the EV3 Brick). The elements are in a specific order, and there can be duplicates.

  For example, you could use a Numeric Array to specify the Set of Colors for the Color Sensor block in Compare – Color mode.

  A Numeric Array displays as a list of numbers separated by semicolons (“;”). The entire list is enclosed in square brackets (“[ ]”). Some examples are shown below.

Numeric Array	Length (number of elements)
[ ]	0
[3]	1
[2; 3; 5]	3
[0; -0.2; 845.25; 5; 5; 5]	6

  You can create an array, add elements, access individual elements, and measure the length of an array using the Array Operations block. You can also create an array with the Variable block.

- LOGIC ARRAY

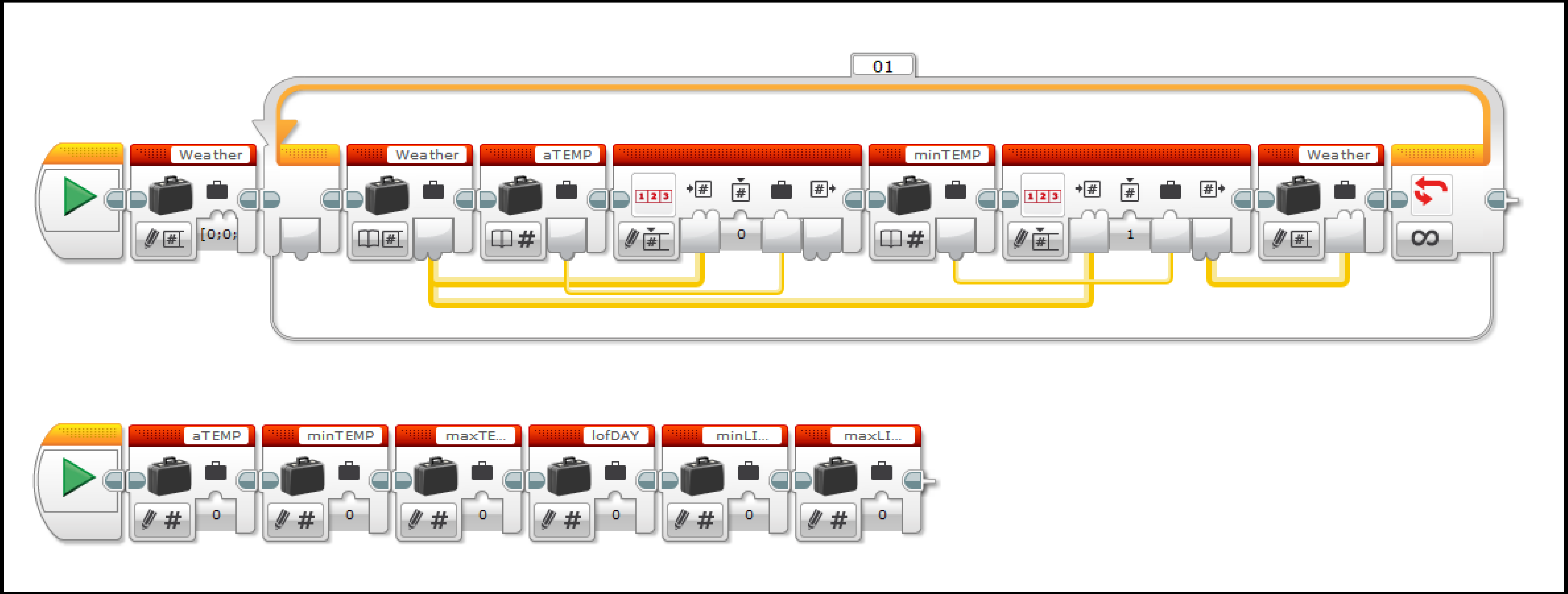
  The Logic Array type represents a list of Logic values. This is similar to the Numeric Array type as described above, except that each element in the array is a Logic value and can only have the values True or False.

EXAMPLE FOR AN ARRAY

- Let’s imagine we’re going to build a little weather station in our classroom
- We’re going to use the following variables to track data:
  - Average TEMPERATURE, aTEMP
  - Maximum TEMPERATURE, maxTEMP
  - Minimum TEMPERATURE, minTEMP
  - HOURS OF DAYLIGHT per day, lofDAY
  - Maximum LIGHT LEVEL, minLIGHT
  - Minimum LIGHT LEVEL, maxLIGHT

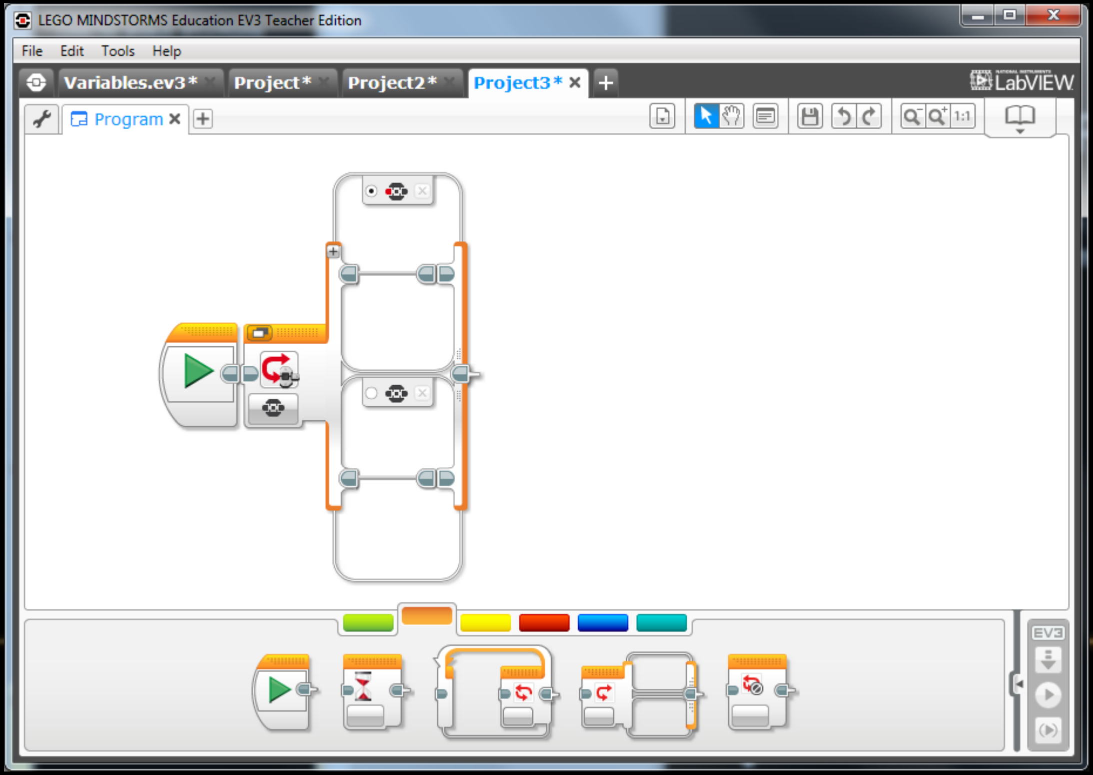
ARRAY OPERATIONS

- Here you see an VARIABLE ARRAY being defined with 6 index points [0;0;0;0;0;0]
- We also see the six VARIABLES for the weather data stored separately
- Inside the LOOP we see the ARRAY read, then we read aTEMP and write it into the ARRAY at index 0, and we read minTEMP and write this to index 1.
- Finally we write the entire array back into the WEATHER array with the new information.



CONDITIONAL PATHWAYS

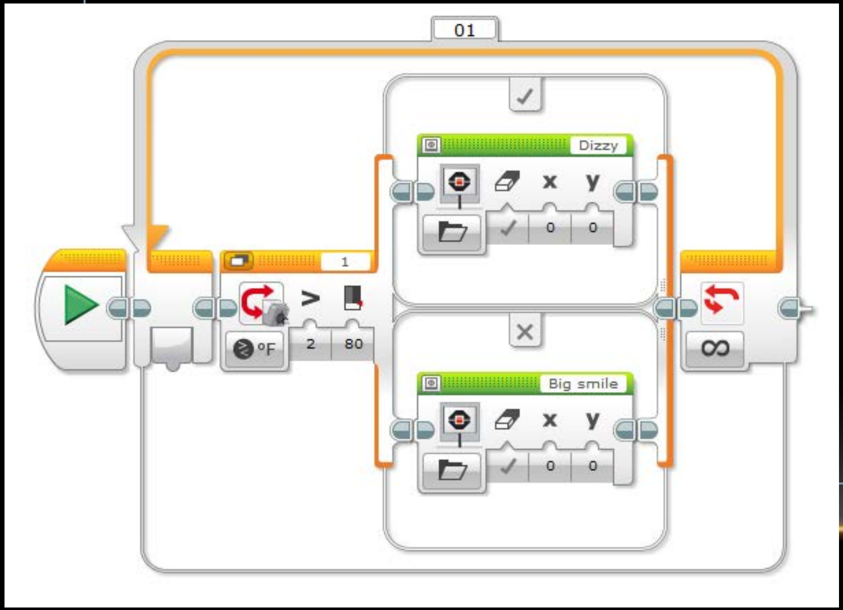
- SWITCHES allow you to pick between different pathways in a program, depending on the value of a variable or sensor value.
- SWITCHES are found in the orange FLOW CONTROL option tab

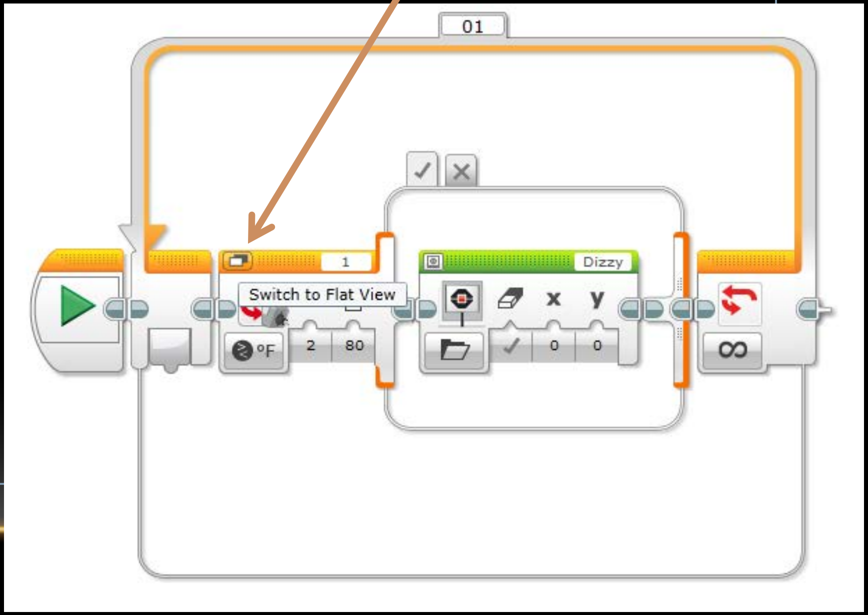


LEGO MINDSTORMS Education EV3 Teacher Edition
File Edit Tools Help
Variables.ev3* Project* Project2* Project3*
LabVIEW
Program
EV3

SENSOR CONTROLLED BEHAVIOR

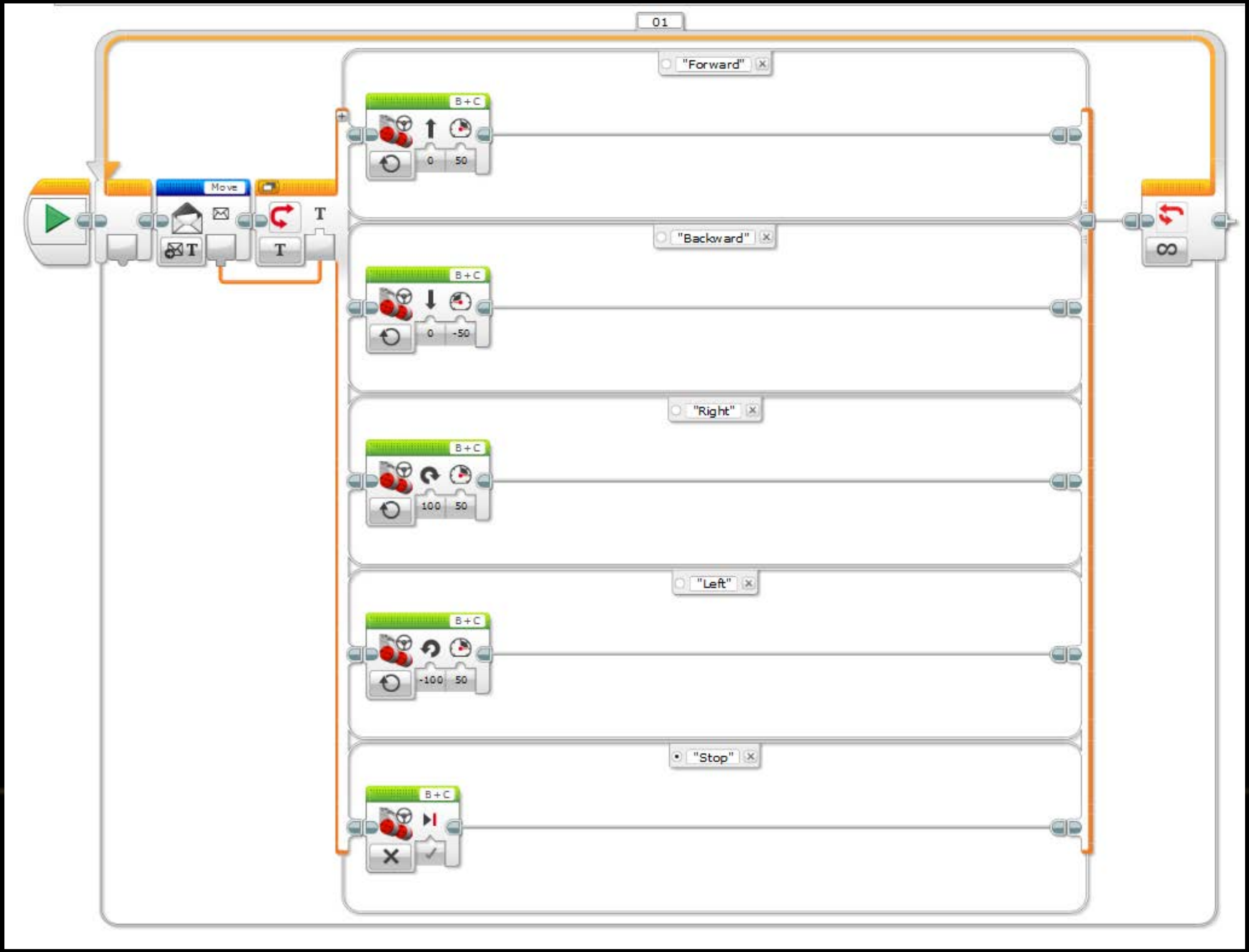
- Here we are using a SWITCH, almost always found inside of a LOOP, to use a TEMPERATURE sensor to determine if the DISPLAY shows a dizzy face or a smile
- On the left you see what is called the expanded view and on the right is the tabbed view. You switch between the two by clicking on the small window at the upper left corner of the SWITCH block. You then select the tabs to see the different options.





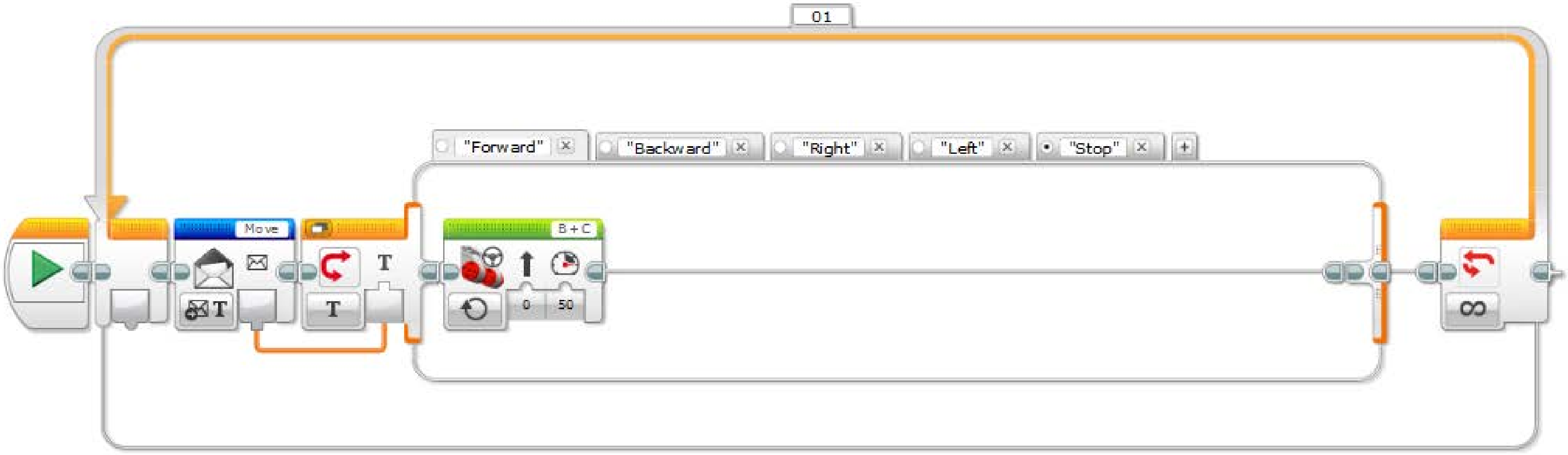
BUILDING A MULTIPLE OPTION SWITCH

- In this example, we have a robot that can take in a Bluetooth message and use the incoming TEXT to determine its behavior (Forward, Backward, Right, Left, Stop)
- Shown here in expanded view



SAME PROGRAM SHOWN IN TABBED VIEW

- Here is the same program, but in TABBED view. Notice how much smaller it is.
- Also note that the “Stop” tab is selected (black dot) this indicates which option I’ve set for the default. You should always have a default that does the least damage…



TIMERS

- Timers are used in programs to count time in seconds
- They don't give "real world time", nor are they formatted in hours:minutes:seconds
- They are handy because if you rely on WAIT options, you will eventually mess up due to time to actually run program elements.

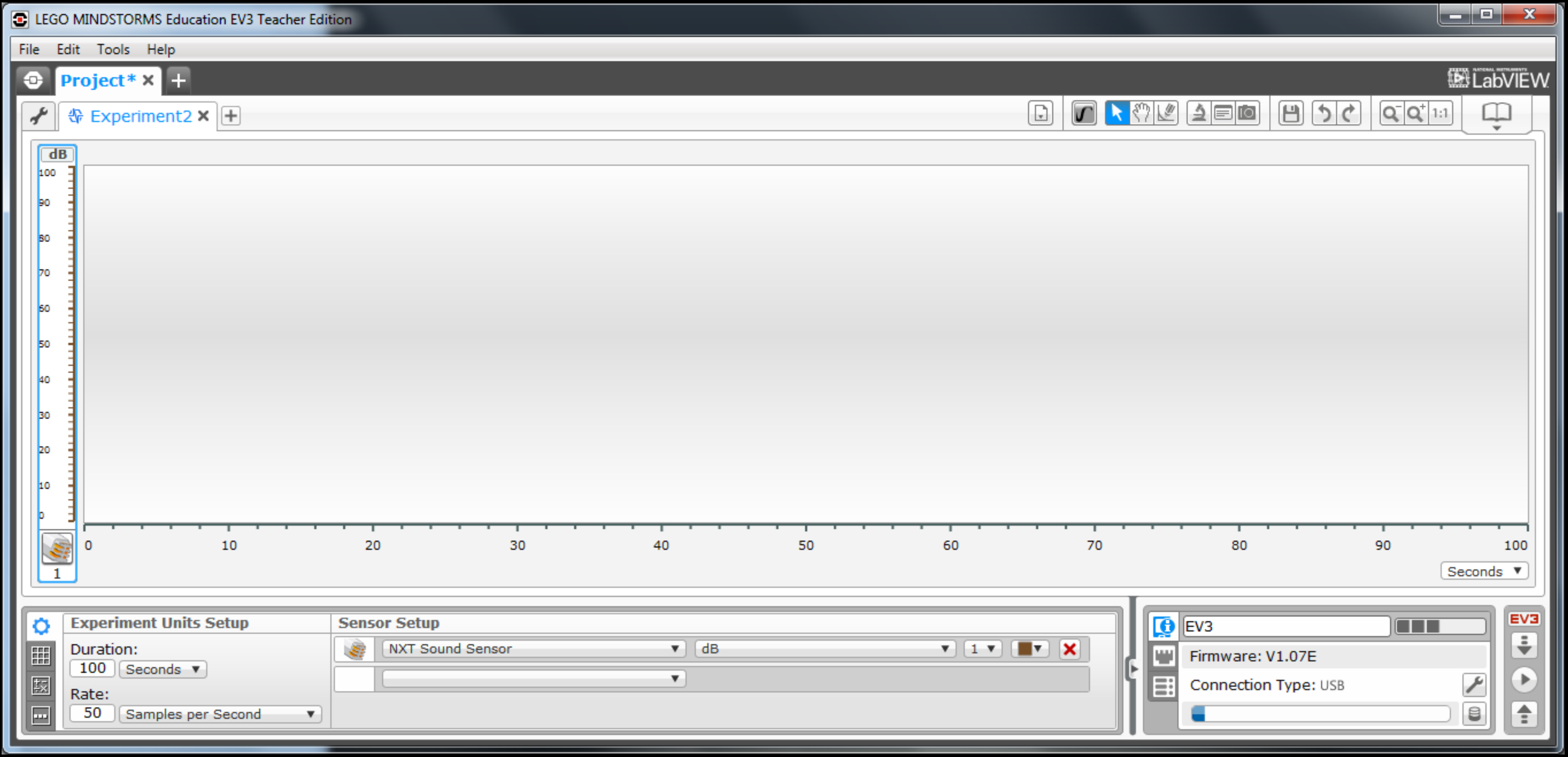
- If you ran a loop for 10,000 times, with each loop counting 1 and WAITing 1 second, you would think you would end 10,000 seconds later… but it takes time to process the program and that will translate to 10,000 seconds PLUS the time to calculate.

- Using a TIMER, you can start the TIMER and stop your program 10,000 seconds later and it should be within a few milliseconds of what you wanted.

DATA LOGGING

- With data logging you can use the EV3 (and the NXT) to collect data.
- For each sensor attached to the EV3, you can specify how often you want to sample
  - Take samples every:
    - 1/1000 of a second
    - 0.5 seconds
    - 180 seconds (3 minutes)
    - 3600 seconds (1 hour)
    - 86400 seconds (1 day)
    - And everything in between
  - You can take multiple inputs for each sample time
    - Check Light values and Temperature values at the same time

- Shown below is a PROJECT with an EXPERIMENT open
- Note that an NXT Sound Sensor is showing on PORT 1 (that is what is plugged into the EV3)
- On the lower left are the variables to set how long the sampling will occur (seconds or minutes) and how many samples will be collected per second (OR how many seconds between samples)
- You can EITHER click the Oscilloscope button above or hit the RUN button



Experiment Units Setup

Duration: 100 Seconds

Rate: 50 Samples per Second

Sensor Setup

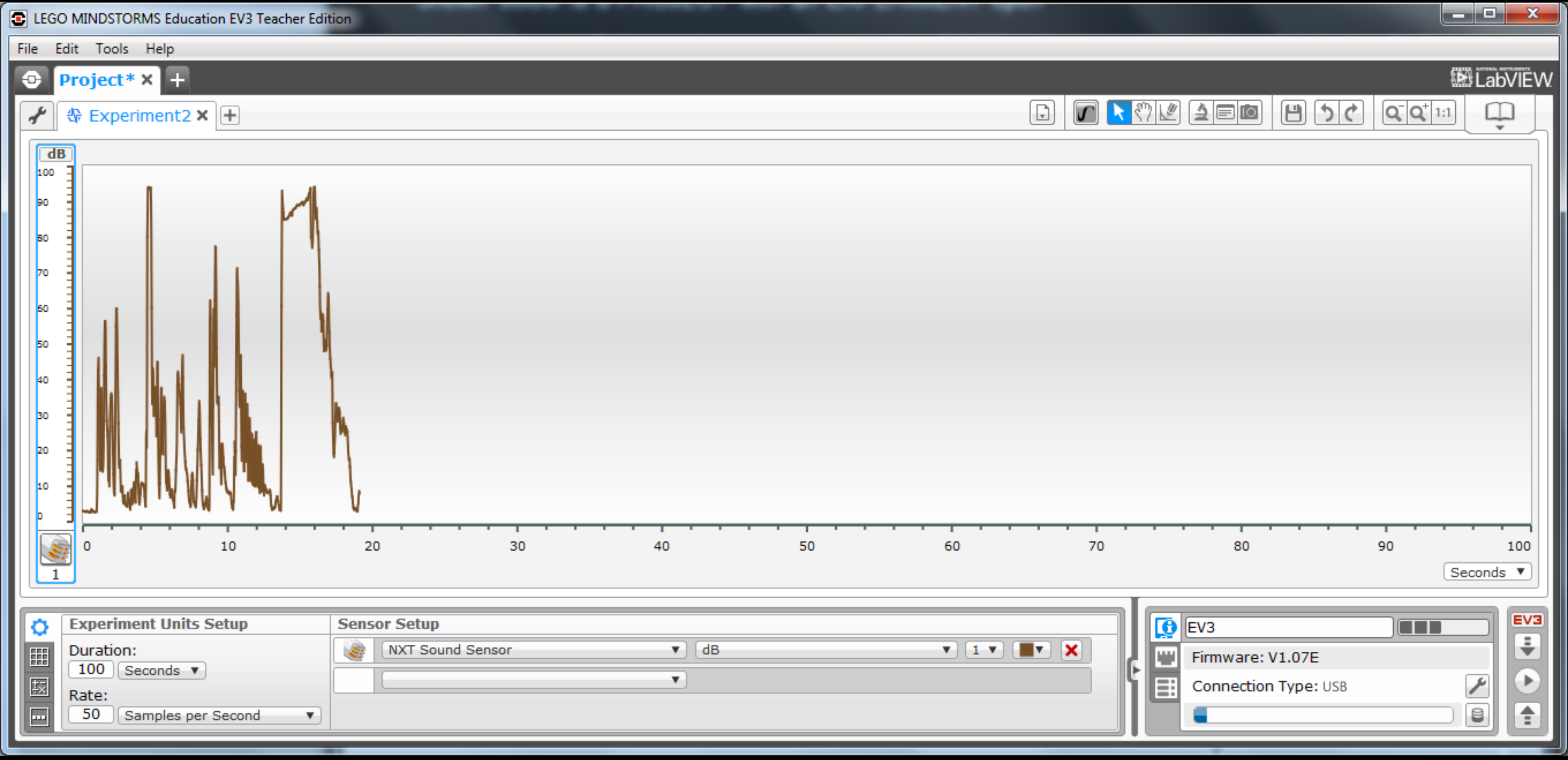
NXT Sound Sensor | dB | 1

EV3

Firmware: V1.07E

Connection Type: USB

- This shows one sample set, notice the color for the displayed data is set next to the sensor. You can set different colors for different sensors OR different data sets for the same sensor.



Experiment Units Setup

Duration: 100 Seconds

Rate: 50 Samples per Second

Sensor Setup

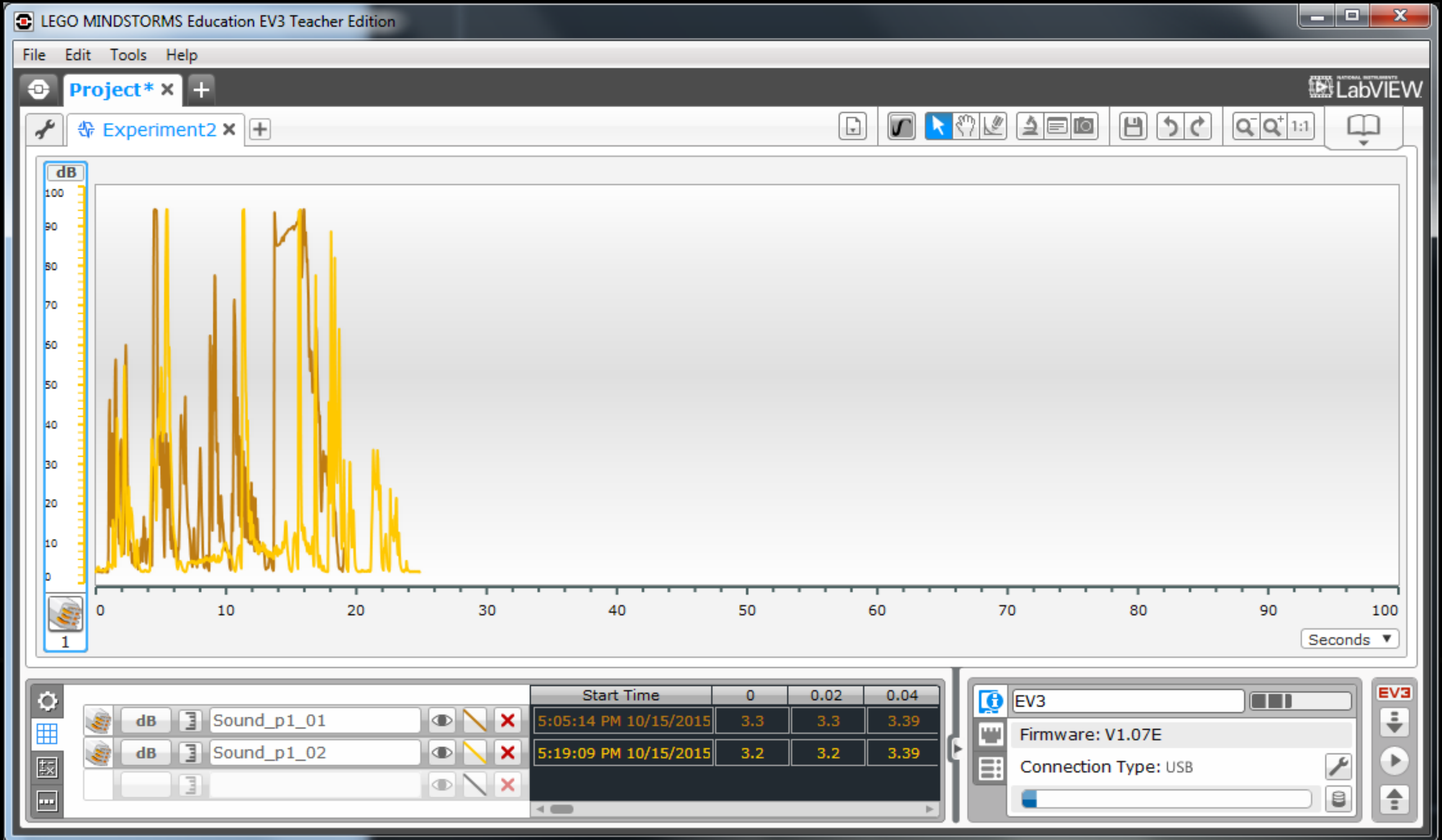
NXT Sound Sensor | dB | 1

EV3

Firmware: V1.07E

Connection Type: USB

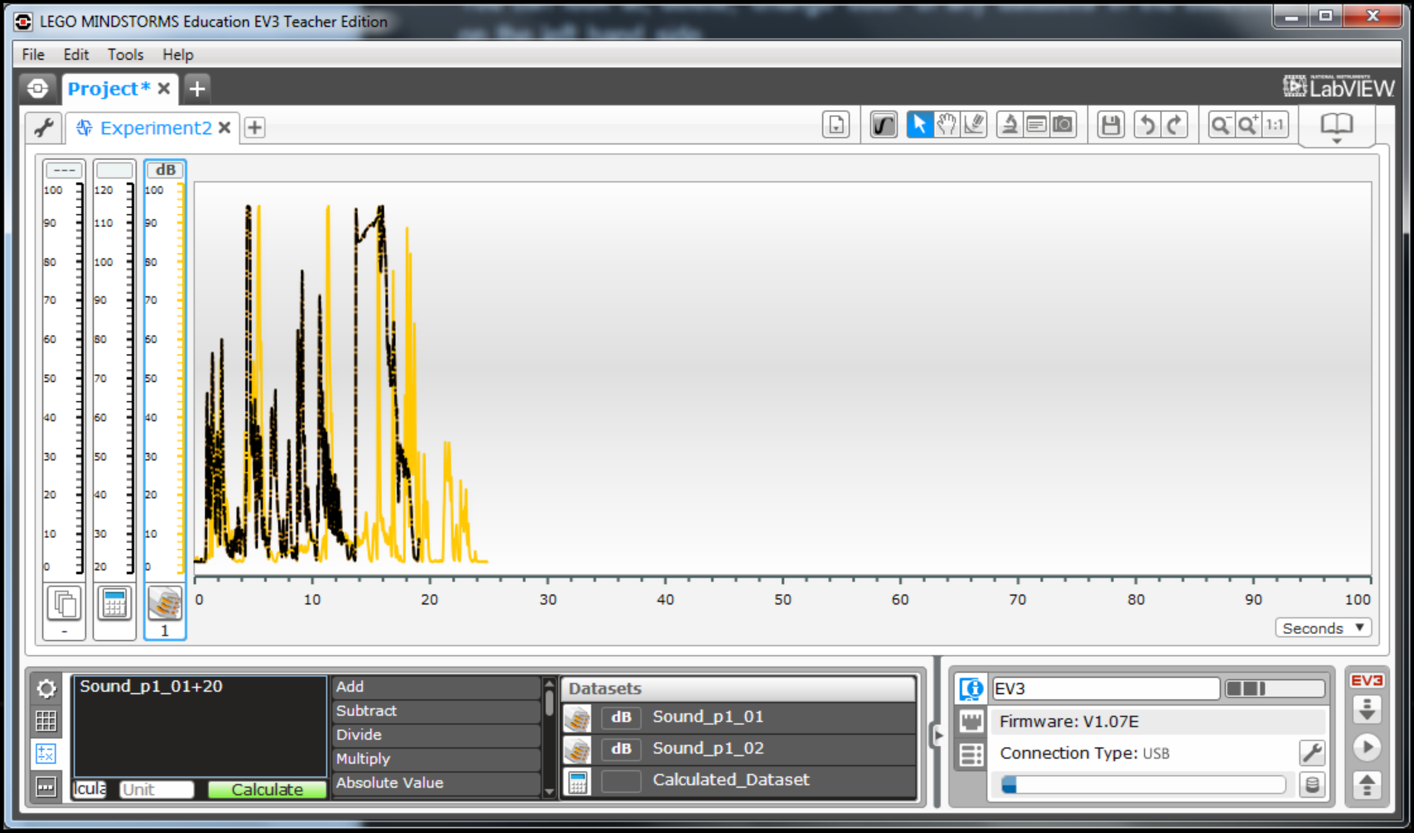
- You can look at, delete, change color of any datasets in the collection using the second tab down on the left hand side



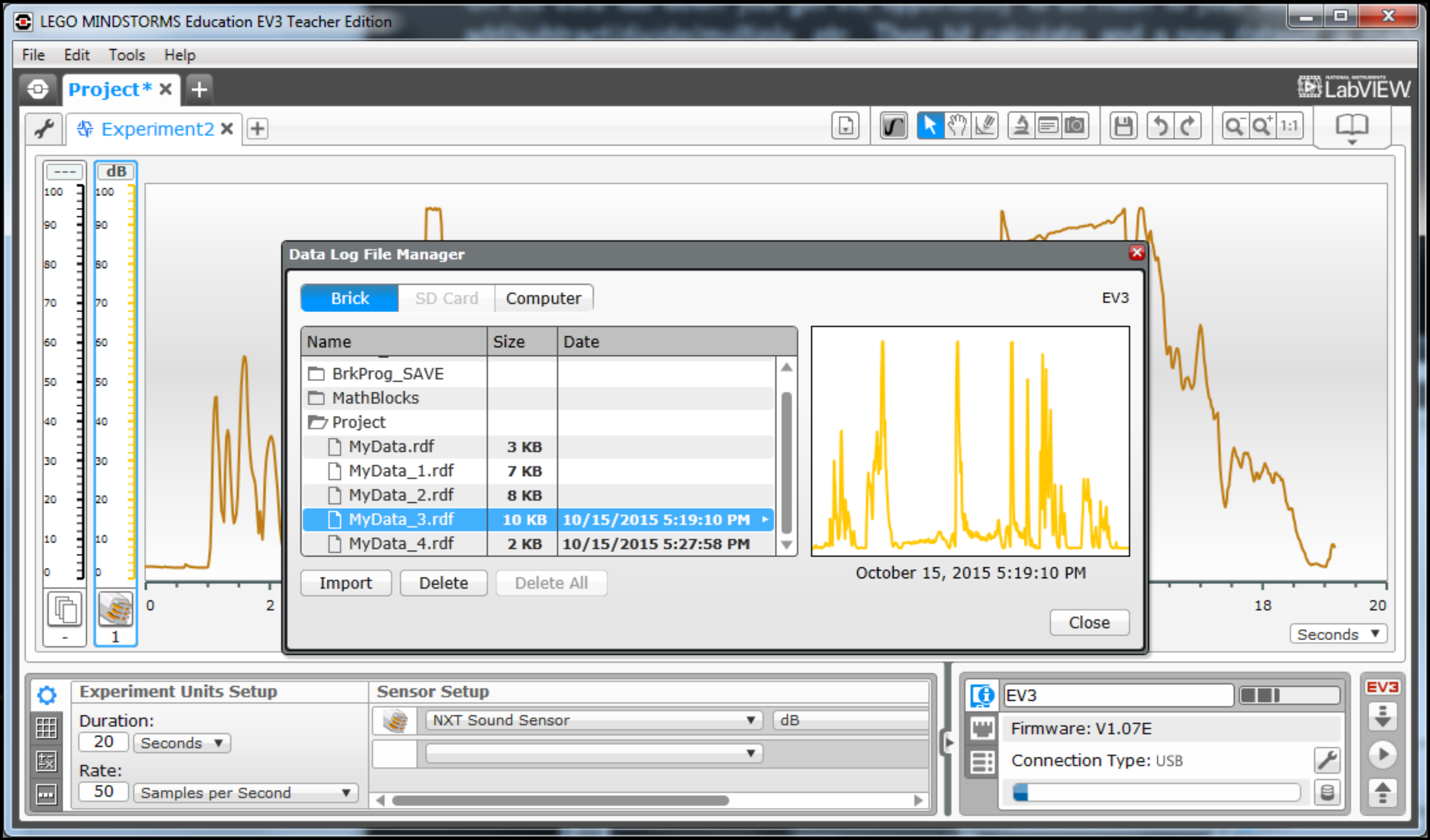
			Start Time	0	0.02	0.04
dB	3	Sound_p1_01	5:05:14 PM 10/15/2015	3.3	3.3	3.39
dB	3	Sound_p1_02	5:19:09 PM 10/15/2015	3.2	3.2	3.39

EV3
Firmware: V1.07E
Connection Type: USB

- On the third tab down you get the opportunity to do math to your dataset. Use the dataset and then add/subtract/divide/multiply, etc. Then hit calculate and a new dataset is made and will be visible in the chart as well as in the list of datasets.



- On the EV3 panel in the Lower Right, you can select the upload button. Then select the name of the Project (Project in this case) and then find the dataset you want to import.



Name	Size	Date
BrkProg_SAVE		
MathBlocks		
Project		
MyData.rdf	3 KB	
MyData_1.rdf	7 KB	
MyData_2.rdf	8 KB	
MyData_3.rdf	10 KB	10/15/2015 5:19:10 PM
MyData_4.rdf	2 KB	10/15/2015 5:27:58 PM

October 15, 2015 5:19:10 PM

Duration: 20 Seconds
Rate: 50 Samples per Second
NXT Sound Sensor dB
EV3
Firmware: V1.07E
Connection Type: USB

DATA LOGGING

- The main difference with Data Logging is that you can save the information and bring it back to the computer
- This allows you to do post-collection data manipulation right in the classroom