Jade Robot™ Scratch Programming Language Outline

Myke Predko

Last Updated: December 30, 2014

License and Warranty

This document and code was written for the Mimetics Jade Robot™ and follow on products.

This document and code is considered Mimetics Proprietary and may not be released outside of Mimetics except by permission of Mimetics, Inc.

Software Compatibility

The script language described in this document was written to be supported by:
Jade Support 0.9.9.26 or later
Robot Software Release 42 or later
Robot Tokenizer Version 0.11.17 or later
XML Processor 0.1.5 or later
_start.s defined as _main_10.script or later

Conventions, Options and Selections

Example code will be put in monospace font like:

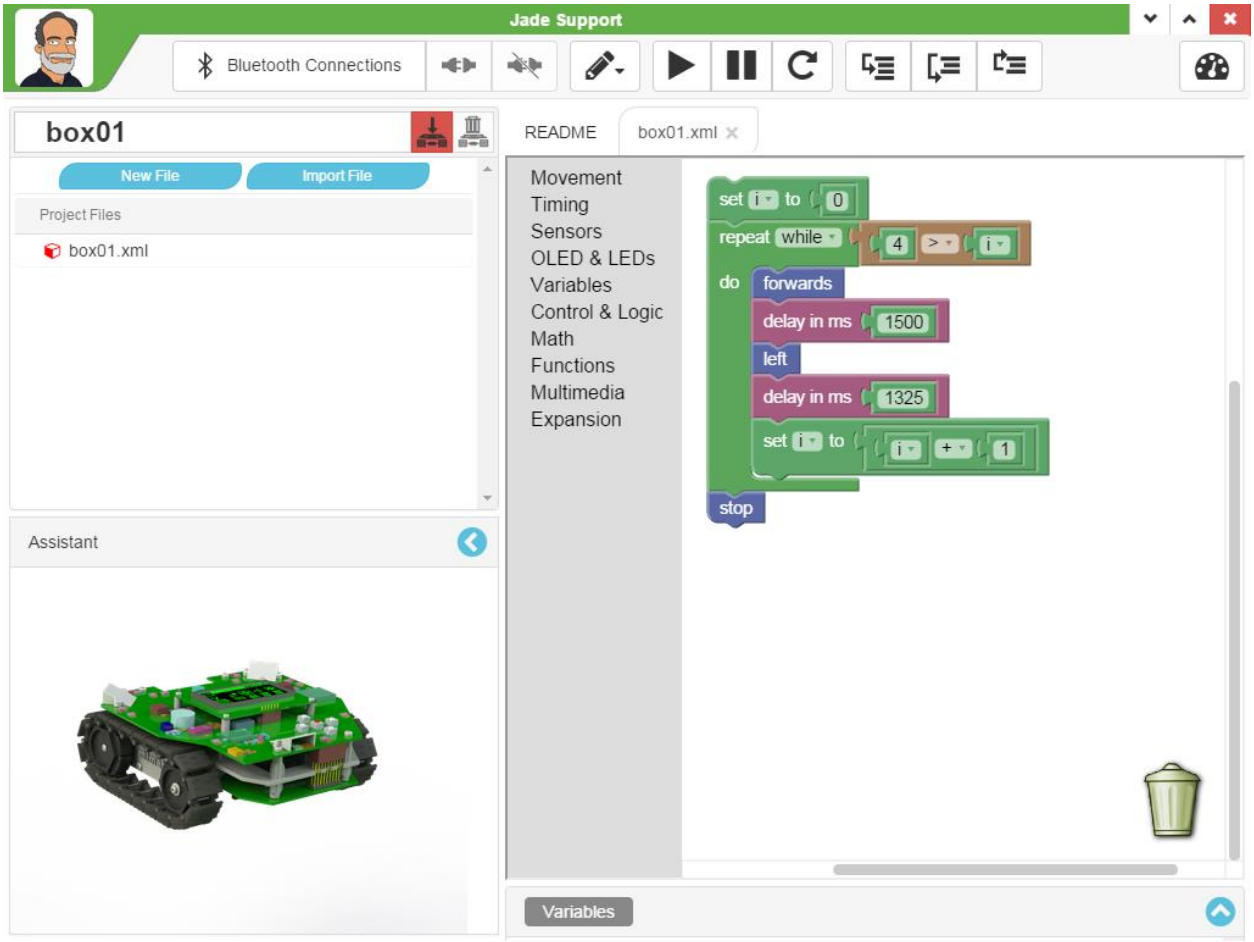
```
A = B + C
```

In the language definition, there are a number of instances where there are optional parameters or multiple parameters for the same task. To make these situations more obvious, the following convention is used:

[] – Optional parameter
| - One parameter or another
... – Previous parameter can be repeated
<none> - indicates that nothing is a possible option

Overview

The introductory and, arguably, the most efficient method of programming the Jade Robot™ is to use the version of the "Scratch" programming language described here. The Scratch programming "authoring tool" was first developed by the MIT Media Lab's Lifelong Kindergarten group in 2006 and has gained prominence as an introductory approach to teaching programming and the thinking skills that go along with it. "Jade Scratch" has been integrated into the "Jade Support" Integrated Development Environment (IDE) for the Jade Robot (see screen shot below), allowing programs to be created by dragging and dropping different programming elements into it.

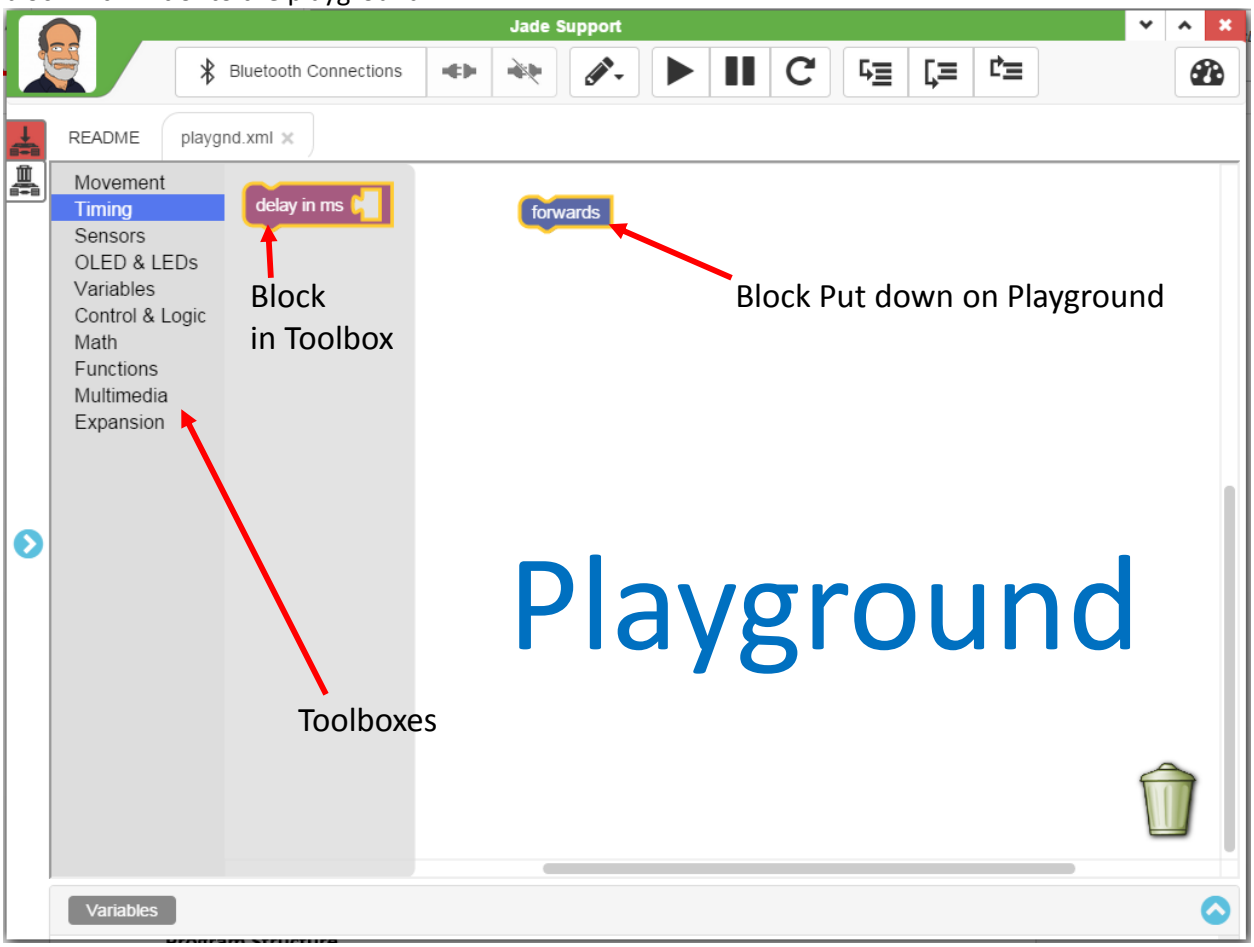


While graphical, the Jade Scratch is a structured, procedural programming language designed to train users for text based programming. The Jade Scratch language does not provide all the programming "Blocks" of the original MIT Scratch Programming language; these were found to be different from traditional programming constructs and not easily transportable to them. Along with this, additional blocks have been provided that are specific to the Jade Robot's functions. Finally, only one data is provided with the language: integers (the original Scratch programming language has multiple data types including Booleans and Strings). These changes to the Jade Scratch language over the original MIT Scratch version were to minimize the amount of background learning required of the user to get started programming the Jade Robot.

Jade Scratch is based on the Google "Blockly" development tool and has been modified from the basic standard in a number of areas. As noted in the previous paragraph, a number of blocks have been removed from the Scratch/Blockly set to provide a similar set of statements to traditional structured programming languages as well, how comments are implemented have been changed as well as adding a number of blocks that provide functions that are specific to the Jade Robot. Jade Scratch will continue to be updated as more user feedback is received as well as fixing any found bugs.

Creating Programs

Jade Scratch Programs are created by clicking on a tool box name to open them and then dragging a block within it onto the playground:

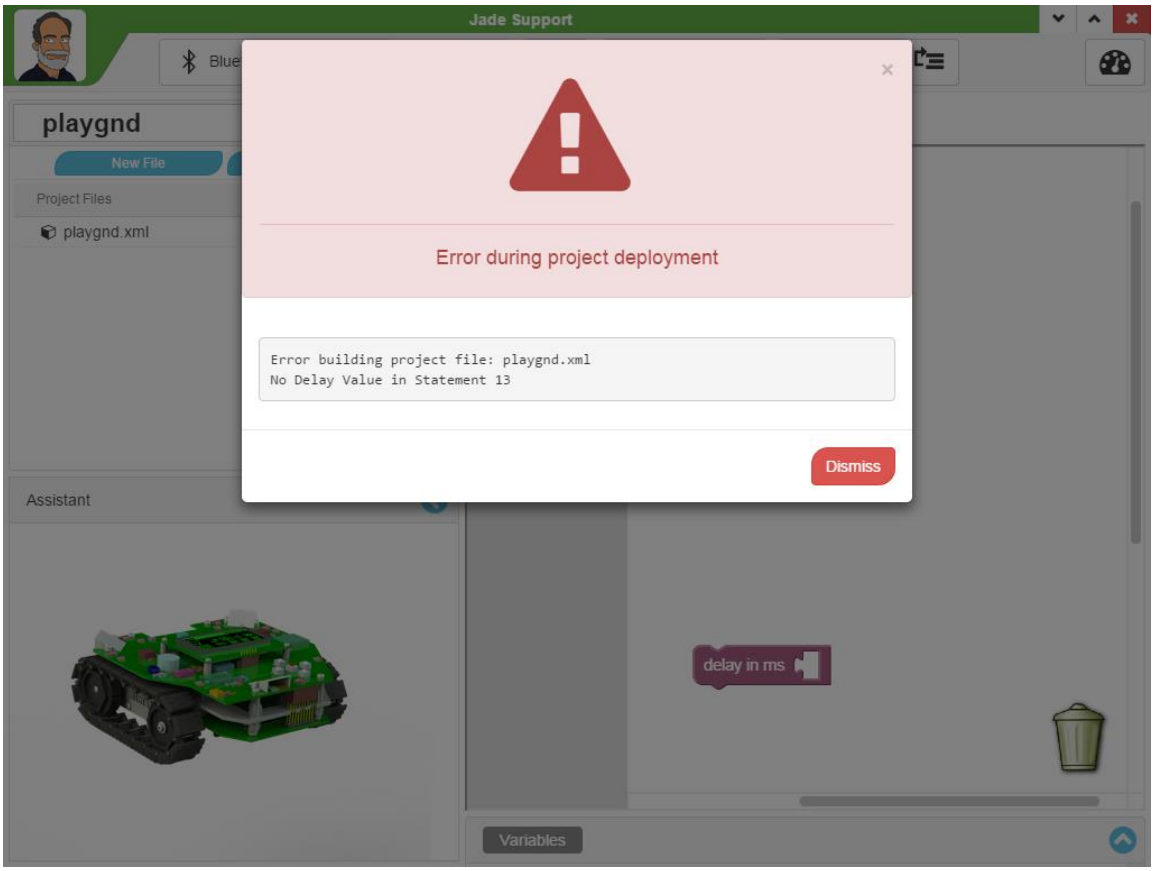


Errors and Warnings

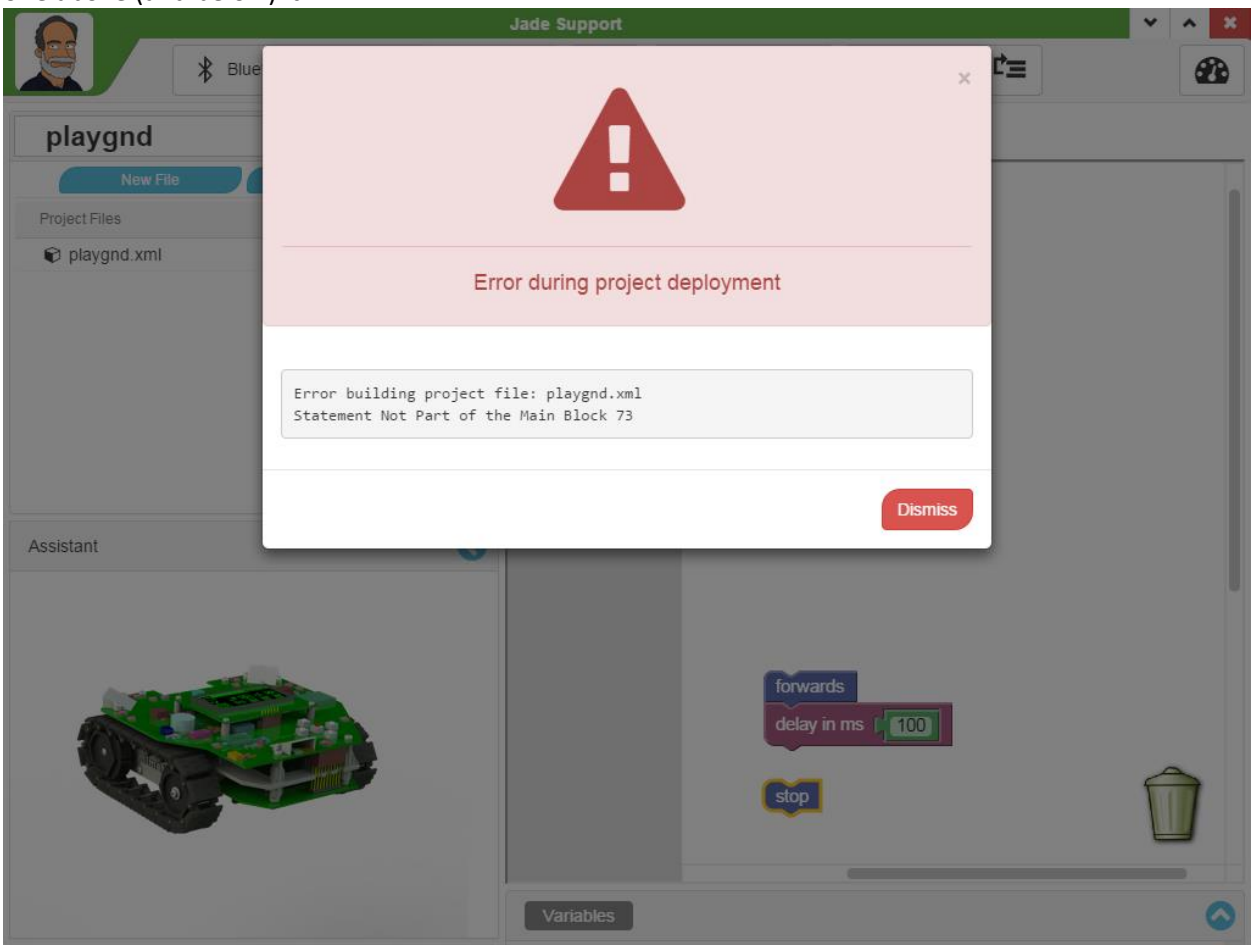
An important philosophy with regards to the Jade Robot is the desire for simplicity. This can be seen in the single data type and is also in place for errors, especially in regards to Jade Scratch. One of the great aspects to Jade Scratch is that there are only three types of build errors that can be encountered and no warnings. The reason for this approach is avoid overwhelming a new programmer.

The three error types consist of:

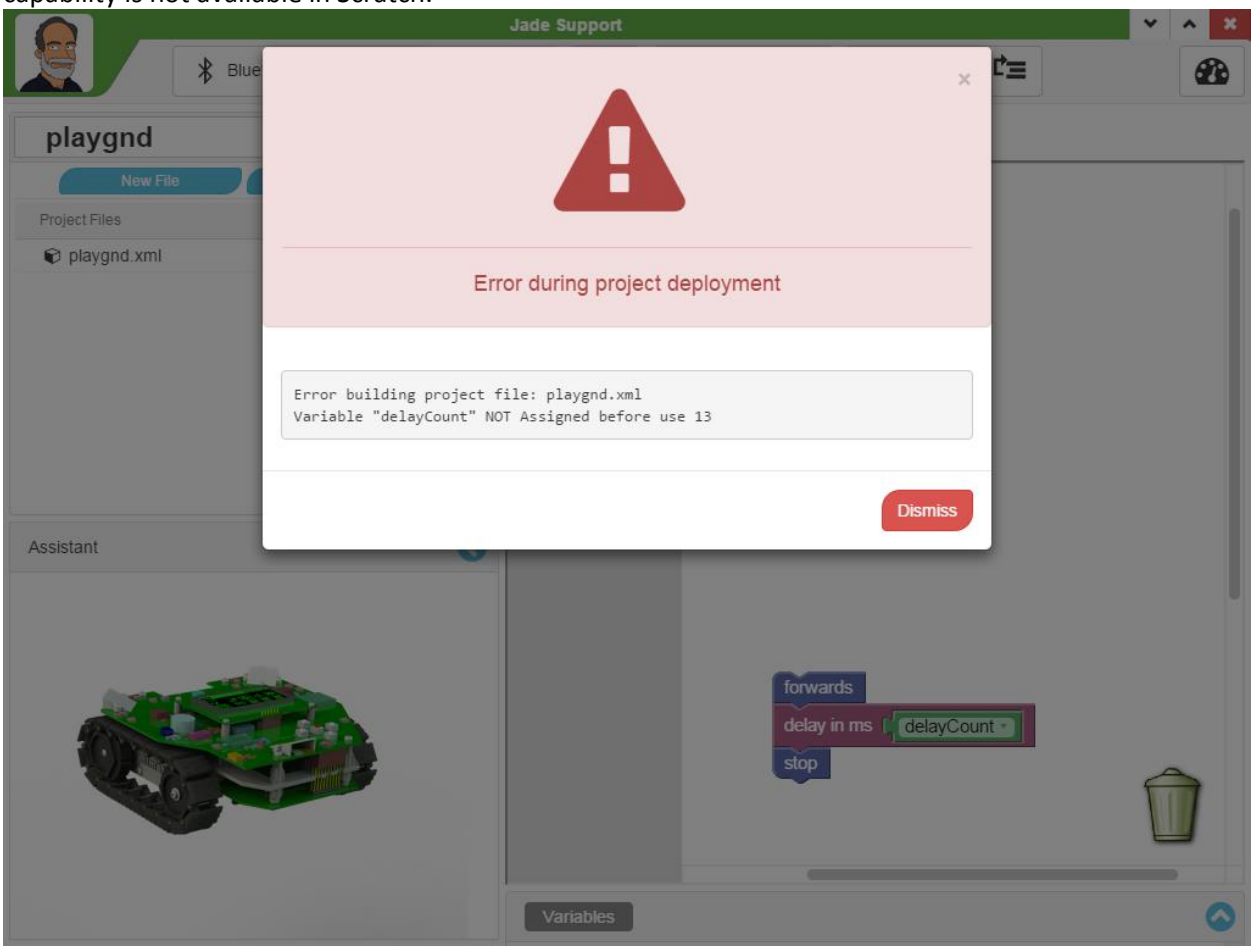
1. Not filling in all the available boxes in blocks. Certain blocks require data input for their operation, failing to put data blocks into them will result in the error shown in the screen shot below. Remember, you should not see the white background of the Scratch "playground" in any of the blocks.



2. Not connecting all the statement blocks in a Scratch Program, like in the example below. With the exception of function and Subroutine declarations, every block needs to be connected to the one above (and below) it.



3. Using a variable before you have initialized it. If you are experienced at all in programming, you will know that this is a trick that occasionally trips up the best of us; using a variable for the first time that has not been given an initial value. In Jade C, variables cannot be used without a previous declaration and when they are declared, they are given an initial value of zero (0) – this capability is not available in Scratch.



With respect to run time errors, if the Jade Robot stops or behaves unexpectedly, remember to check the display on the Jade Robot itself. All six of the green LEDs should flash, indicating that a run-time error has occurred as well as display a box on the display with an error message that provides you with failure information:

Debugging

Debugging a Scratch program consists of “Pausing” execution of the program (this is a “halt\r” command to the Jade Robot) and then single stepping or examining the contents of variables. To resume execution, click on the “Run” icon on Jade Support. Unfortunately, Scratch does not lend itself to breakpoints and other sophisticated methods of software debugging.

Debugging and the capabilities listed above can be implemented with a Bluetooth or USB connection.

Data Types

As noted elsewhere, “Integers” is the only data type allowed in Jade Scratch:

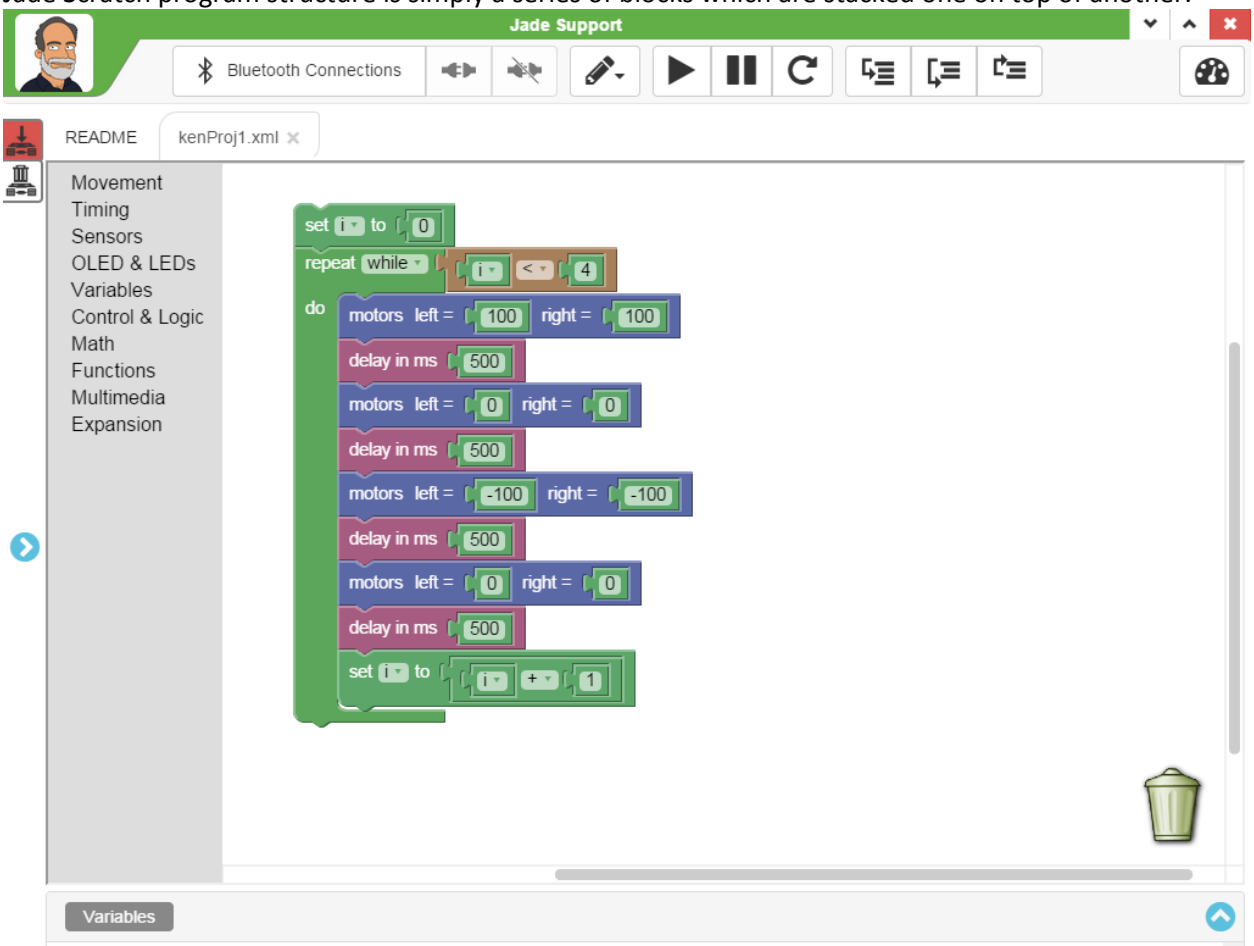
- signed 32 bit integers (ranging from -2,147,483,648 to +2,147,483,647)

The Jade Robot uses a 32 bit ARM Cortex M4, so there is no speed penalty defaulting to the 32 bit integer data type.

Strings can be specified when writing text to the Jade Robot’s display but they cannot be saved as a data type. This is also true for arrays.

Program Structure

Jade Scratch program structure is simply a series of blocks which are stacked one on top of another:

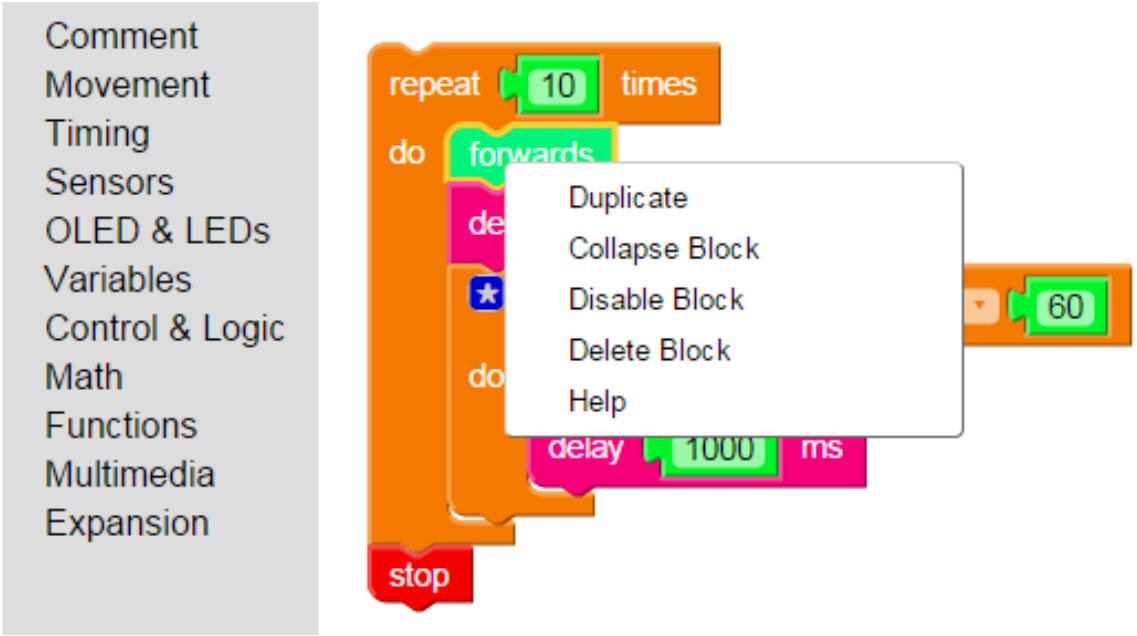


There is no “main” function and, as noted above, the only requirements in a program is that there cannot be any unfilled areas in blocks, each block must be connected to the ones above and below and variables must have a set value as their first use.

Functions are placed outside the mainline and follows the same rules.

Block Options

Options for the blocks can be accessed by right clicking the block in Windows or Control-click in a Mac with the following block of options appearing:



"Duplicate" copies and pastes the current block as well as any parameters as well as any enclosed expression and operation blocks onto the Jade Scratch Playground. If a statement with enclosed statements (such as "repeat" in the image above) is duplicated, then just the statement is duplicated, not the enclosed statements.

"Collapse Block" will reduce the current block to a single line with any enclosed statements hidden. The purpose is to make the full program easier to read through by minimizing statements which are fully understood and not important to going through the program flow.

"Disable Block" changes the appearance of the block and any parameters including any enclosed expression and operation blocks as well as any enclosed statements. When the program is built, the disabled blocks will be ignored. This feature has been found useful in situations where there is a need to put sample blocks on the playground as reference. **NOTE:** if enclosed expression or operation blocks are disabled, the program build will fail with an error.

"Delete Block" will allow for the deletion of the current block and any enclosed expression and operation blocks as well as any enclosed statements. This is often faster and easier than dragging the blocks to the Trash Can.

"Help" opens a link to this document.

In previous version of Jade Support, "Comments" could be added at this point, but this feature has been removed in favor of the explicit "Comment" block that can be added to the program.

Statements

The Jade Robot script language follows the statement conventions set out in C and Java; that is to say that variable declarations, assignment statements as well as subroutine calls all end in a semicolon (“;”) and function entry points along with conditional execution statements end in an open curly brace (“{“).

Statements are not line based – they start on the character after the previous statements end character and continue on to the next end character. This means that multiple statements can be placed on the same line, or a single statement can run over multiple lines.

Labels

Labels are used for variable, function and subroutine names. Labels can be up to 30 characters in length. They follow normal conventions and can comprise of the following characters:

- “_” (Underscore), anywhere in the label
- “a”-“z” (Lower case alphabet characters), anywhere in the label
- “A”-“Z” (Upper case alphabet characters), anywhere in the label
- “0”-“9” (Numerics), anywhere in the label EXCEPT the first character

Other than specifying the label “item” for variables, there are no reserved words that are cannot be used in Jade Scratch

Valid labels include:

- counter
- test01
- _userID

Invalid labels include:

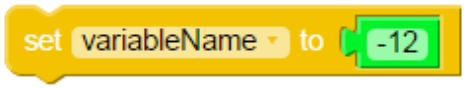
- 7counter
- _@@
- LabelNameLongerThanThirtyCharactersLikeThisOne

Reserved Words

There are no reserved words (with the exception of “item” for variables, as noted in the previous section) in Jade Scratch.

Variable Declarations and Assignments

Variables are automatically declared when they are used in a “set” assignment statement. If a variable name is repeated in a “set” assignment statement, it is assumed to be the same variable used earlier.



It must be noted that a variable must be “set” – if a new variable name expression block is encountered before a corresponding “set” statement, then there will be an error flagged during the build/transfer to Jade Robot.

The default variable name is “item” and it should be noted that this name cannot be used. Along with “item” variables starting with “_loopT” and “_loopC” cannot be used; these variables are used for the “Repeat” block.

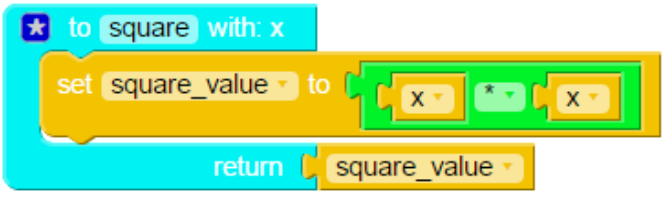
File Types

Jade Scratch programs are actually XML files which describe the program and is why when you look at a Jade Scratch file name in Jade Support it ends in “.xml”. When the executable version of the program is loaded into the Jade Robot, it is given the extension “.s”. Other files that can be included in the project with the Jade Scratch XML file include:

- .bmp – bitmaps used to display images on the Jade Robot’s OLED
- .wav – audio files used to output polyphonic sounds from the Jade Robot

Functions and Subroutines with Local Variables

Jade Scratch functions and subroutines work much the same way as their countparts in traditional programming languages; parameters can be passed to them (and are saved as local variables), code can execute within the function accessing either local or global variables and a value can be returned as shown in the following example:



In this example, the variable “i" is initialized to the square of the integer “2” using the “square” function which has the “x” passed local variable used to produce the value in the local variable “square_value”.

There is an issue to be aware of and that is the use of local variables which have the same names as global variables – in Jade Scratch, there will be no local variable declaration instead the global variable will be used. The best way to avoid this is to use the function name as a prefix for the local variable as is done here to ensure that it is not confused with a global variable.

With functions, there is no explicit “return” statement, instead, the value to be returned at the end of the function is specified as shown in the example above.

Clicking on the star in the Function declaration will provide you with options to set the input parameter local variable names as well as whether or not there are program statements within the function.

Arithmetic & Logical Expressions and Order of Operations

Arithmetic & logical expressions are built up from blocks and provide a default order of operations. Each arithmetic operation block, like:

Is treated as if it had parenthesis around it in a traditional language like:

```
(value1 * value2)
```

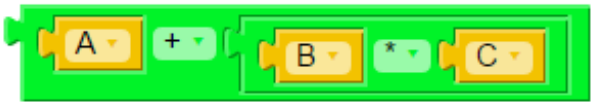
This eliminates the need for concern for any kind of order of operations but may be an issue to some programmers that are used to relying on order of operations for the execution of their statements. For example, if they wanted to execute the expression:

```
A + B * C
```

using order of operations, they would assume that “B” and “C” would be multiplied together first and the product added to “A”. Coding this explicitly, it would be:

```
(A + (B * C))
```

In Jade Scratch, to get the same operation, it would have to be coded as:



because if it was coded as:

the expression would be evaluated as if it were written out as:

```
((A + B) * C)
```

which produces a different result than what was intended.

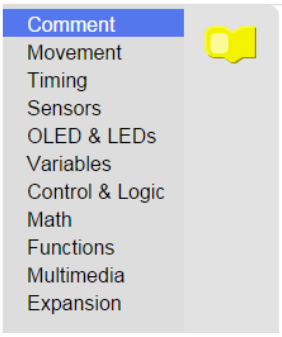
Statements

In Jade Scratch script statement blocks are defined as having an indentation on the top and a bump on the bottom so they can line up with other statements. Expression blocks connect to each other and within statements.

Scratch Block Definitions

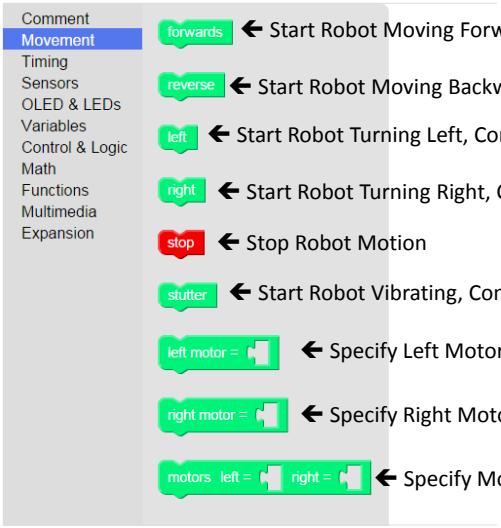
Blocks are grouped according to function within specific toolboxes.

Comment Toolbox



← Add a comment to the program

Movement Toolbox



← Start Robot Moving Forwards, Continues indefinitely

← Start Robot Moving Backwards, Continues indefinitely

← Start Robot Turning Left, Continues indefinitely

← Start Robot Turning Right, Continues indefinitely

← Stop Robot Motion

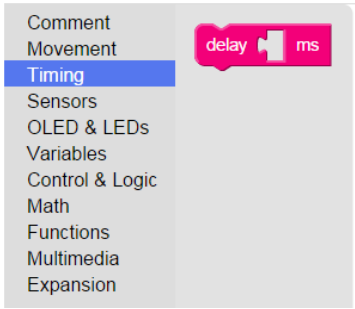
← Start Robot Vibrating, Continues indefinitely

← Specify Left Motor Value (-100 to 100/Negative for Reverse), Continues indefinitely

← Specify Right Motor Value (-100 to 100/Negative for Reverse), Continues indefinitely

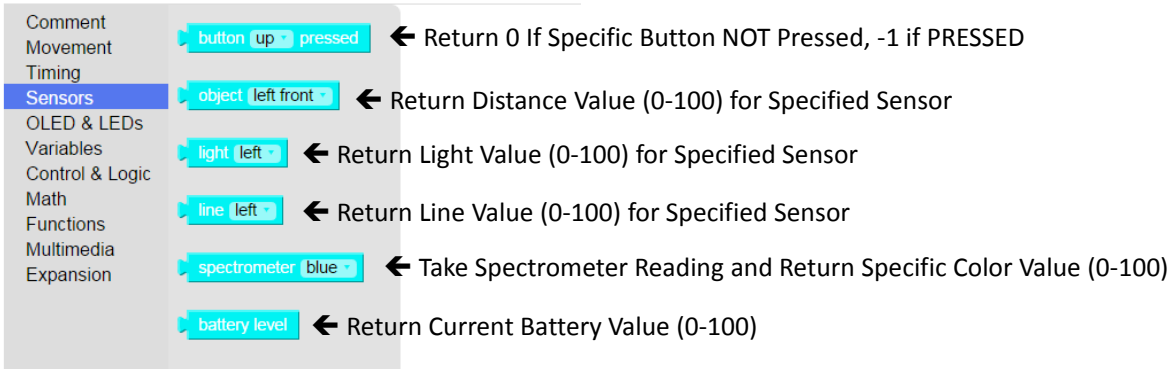
← Specify Motor Values (-100 to 100/Negative Reverse), Continues indefinitely

Timing Toolbox



← Specify Delay Value in ms (1,000ms = 1s). Maxium of 5,000ms

Sensor Toolbox



← Return 0 If Specific Button NOT Pressed, -1 if PRESSED

← Return Distance Value (0-100) for Specified Sensor

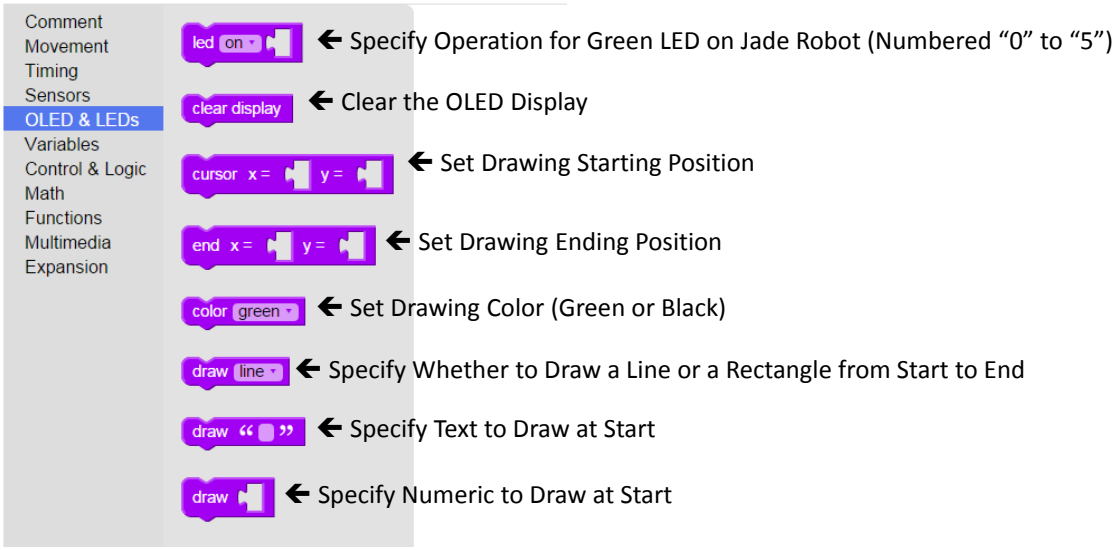
← Return Light Value (0-100) for Specified Sensor

← Return Line Value (0-100) for Specified Sensor

← Take Spectrometer Reading and Return Specific Color Value (0-100)

← Return Current Battery Value (0-100)

OLED & LEDs Toolbox



← Specify Operation for Green LED on Jade Robot (Numbered "0" to "5")

← Clear the OLED Display

← Set Drawing Starting Position

← Set Drawing Ending Position

← Set Drawing Color (Green or Black)

← Specify Whether to Draw a Line or a Rectangle from Start to End

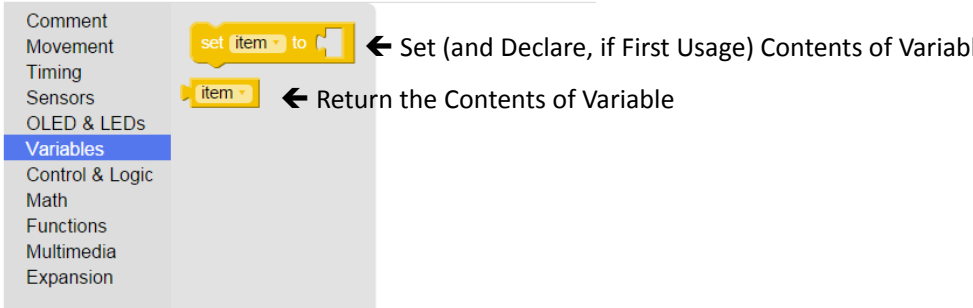
← Specify Text to Draw at Start

← Specify Numeric to Draw at Start

When accessing the LEDs and Display, there are a few things to be aware of:

- The Green LEDs on the top of the Jade Robot are numbered from 0 to 5, starting at the 1:00 position, looking from the rear of the robot, and increasing in a clockwise direction
- The display is 128x64 pixels with 0,0 in the top left hand corner
- After writing to display, "Start" cursor is updated to end of the write allowing continuous text

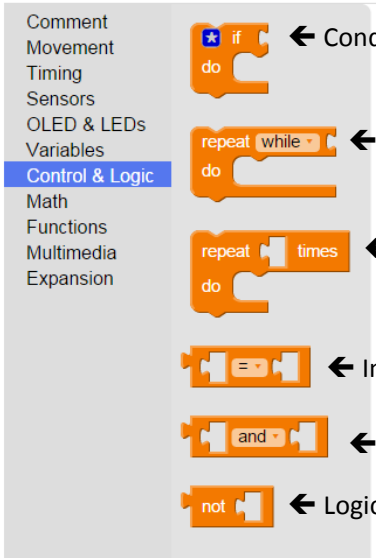
Variables Toolbox



← Set (and Declare, if First Usage) Contents of Variable

← Return the Contents of Variable

Control & Logic Toolbox



← Conditionally Execute Code (also with “else” and “else if”)

← Repeat (Loop) while/until a condition is true

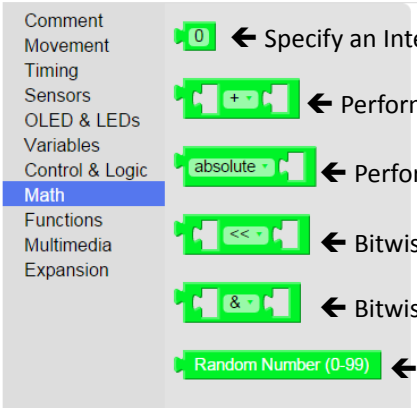
← Repeat (Loop) for a set number of times

← Integer Comparison

← Logical AND/OR

← Logical NOT

Math Toolbox



← Specify an Integer Value (Negatives allowed)

← Perform Arithmetic Operation (Add, Subtract, Multiply, Divide, Modulo)

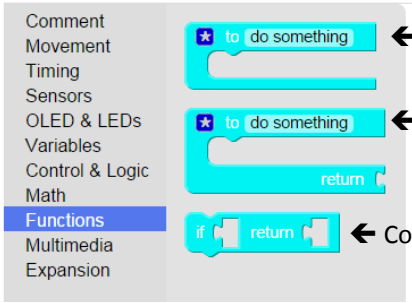
← Perform Absolute/Negation Operation

← Bitwise Shifting

← Bitwise AND/OR/XOR

← Return a Random Number (0-99)

Function Toolbox

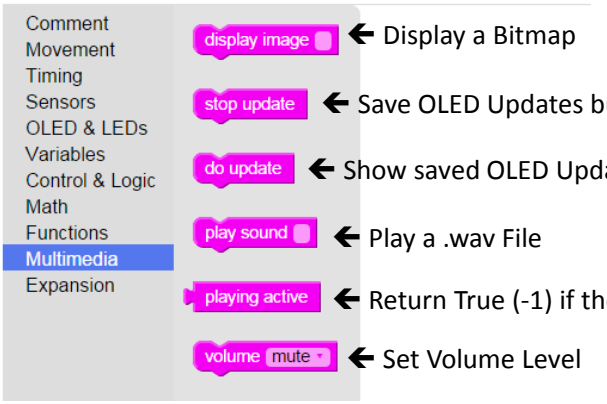


← Define Subroutine

← Define Function (With Return Value)

← Conditional Function Return Value

Multimedia Toolbox



← Display a Bitmap

← Save OLED Updates but do not show on display

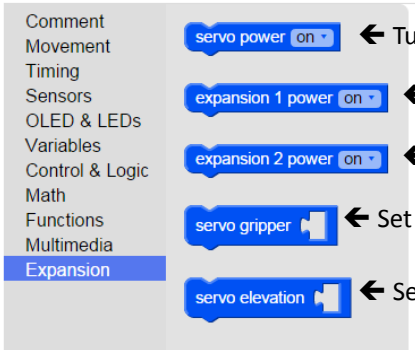
← Show saved OLED Updates and continue to update display

← Play a .wav File

← Return True (-1) if the .wav File is still playing

← Set Volume Level

Expansion Toolbox



← Turn Servo Power On/Off

← Expansion 1 Power On/Off

← Expansion 2 Power On/Off

← Set Gripper Servo (0 – 100)

← Set Elevation Servo (0 – 100)

Supporting Documents

Currently None

Glossary

ASCII – Standard 8 bit character set. See http://en.wikipedia.org/wiki/ASCII

Document Updates

Date	Changes	Author
2014.11.10	Initial Release	Myke Predko
2014.12.30	Updated with 0.4.7 release which consists of: - Updated Block Colours - Added Comment Block - Removed “Comment” from Individual Blocks - Added OLED stop/do Update Blocks - Added Repeat # Times Block - Updated the format to a number of blocks - Added the “Disable” Functionality to the Blocks Add a section on the “Block Options”. Updated the “Variable” description section.	Myke Predko