Itty Bitty City Code Companion – Scratch 3.0

WELCOME!

What the Itty Bitty City Code Companion brings to you!
The Itty Bitty City Code Companion gives even more opportunity to enjoy Itty Bitty City at a more in-depth level. In this Code Companion, you will see how you can change the way projects function, or even make a new project. With Itty Bitty City you can make intelligent, technology-based projects, and with this Code Companion, you can explore even more.

What we are going to do with the Itty Bitty City Code Companion – Scratch 3.0
From the Itty Bitty City user manual, you probably already followed the basic steps to build your projects. And from the downloading Microduino's new mDesigner, you have already had fun using Scratch 3.0 in all your projects. With this Code Companion, we are going to build on a standard Scratch script that comes with the Windmill project and discover and understand how altering a Scratch script will change the way this project or any project works.

What does 'Code' within Scratch 3.0 do?
Part of what you do when you assemble your Itty Bitty City electronics projects using a mBattery, mCookie modules, and sensors, is to make a micro-computer. Like any computer, it needs instructions or code on what to do. Without this code, the micro-computer or CPU/red Core Module has no idea what to do or how to operate.

What does Code look like?
Scratch code is a graphic representation of text-based Arduino IDE code. While text-based code looks like a list of words or sentences on each line, Scratch 3.0 uses color drag and drop blocks to represent portions of required Arduino code needed to make a project function.

Where do these lines of code come from?
Normally, lines of code are written in a program on a computer by a programmer. This program is called an "IDE editor" which has special functions such as checking/compiling the code, setting the communications mode to the project's computer and setting which type of computer board/processor and port is being used.

In Scratch 3.0, each color block represents a specific function. Each function block already has the textual IDE code written for it by a programmer. Each color block has pre-made code built in for the Microduino modules and sensors including those that come with Itty Bitty City. **This makes it easy** to add new functions to your projects.

With the mDesigner editor, a graphical color block is either added or subtracted (via click and drag) from the left side pane area of the mDesigner to the Script area to the right depending on the way you want a project to function.

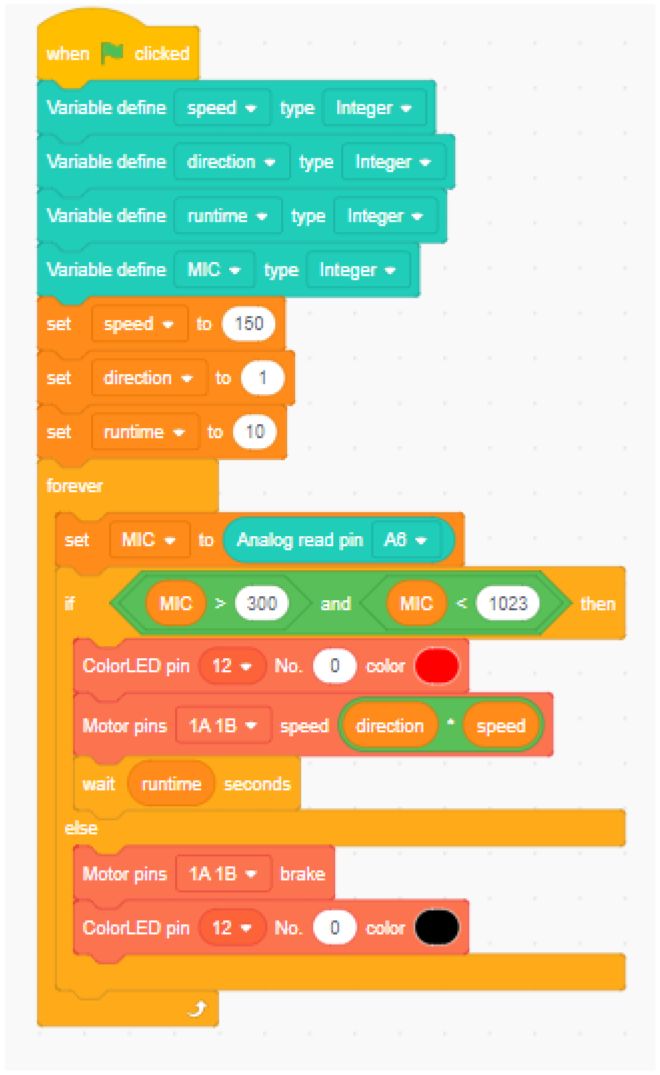
The 'program'.
All the vertical blocks of code stacked together that form a single project are called its 'program' or script. You may have heard people referring to a program that they need to make or modify by a computer.

A program, (an sb.3 file) in this instance, can be opened in your Scratch 3.0 editor, (mDesigner) and can be stored in the Microduino computer CPU (red Core Module), or in the computer you are using.

A word of good advice…. as you make or modify a program, you can save the program as a file on your PC. However, it is best to always make a new name for each file if you save it to your PC so you don't lose the original file.

Let's do something…
OK, so we covered some of the basics. Now let's get some of this going to see better what you've been reading about. We are going to start by exploring the Scratch code for the Itty Bitty City Windmill project. So please get ready by opening the mDesigner program and importing the Windmill sb.3 file. And by now, you have already confirmed the proper operation of the *standard* Windmill project and you are connected to a computer. With that done, let's see how to change the way your Windmill project functions!

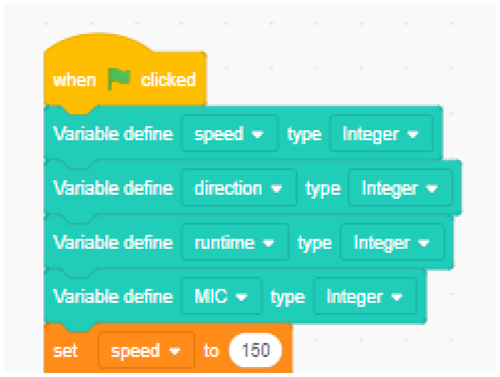
With the mDesigner/Scratch 3.0 editor running and the Windmill file open, (Arduino mode) the standard script looks like the following;



```
when 🏴 clicked
Variable define speed ▾ type Integer ▾
Variable define direction ▾ type Integer ▾
Variable define runtime ▾ type Integer ▾
Variable define MIC ▾ type Integer ▾
set speed ▾ to 150
set direction ▾ to 1
set runtime ▾ to 10
forever
    set MIC ▾ to Analog read pin A6 ▾
    if MIC > 300 and MIC < 1023 then
        ColorLED pin 12 ▾ No. 0 color 🟥
        Motor pins 1A 1B ▾ speed direction * speed
        wait runtime seconds
    else
        Motor pins 1A 1B ▾ brake
        ColorLED pin 12 ▾ No. 0 color ⚫
```

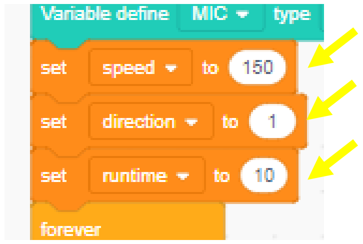
Looking at the above standard Windmill script, notice the multi-color blocks, and notice the individual functions that are associated with each. All the blocks are derived from the various blocks list found in the blocks pane of mDesigner and may be individually dragged over to the script area in a prescribed order, depending on how you want your project to perform.

Next, let's have a closer look at the entire script, but in sections seen below. Under the "when clicked" block, we first have all the variable define blocks taken from the Arduino grouping (left side of mDesigner). We will not be changing any values here.



```
when clicked
Variable define speed type Integer
Variable define direction type Integer
Variable define runtime type Integer
Variable define MIC type Integer
set speed to 150
```

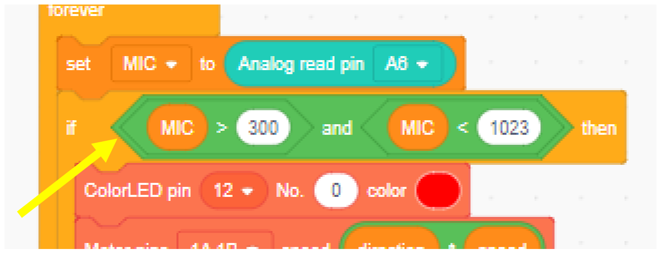
Next notice the three variable blocks that follow, speed, direction, and runtime;



```
Variable define MIC type
set speed to 150
set direction to 1
set runtime to 10
forever
```

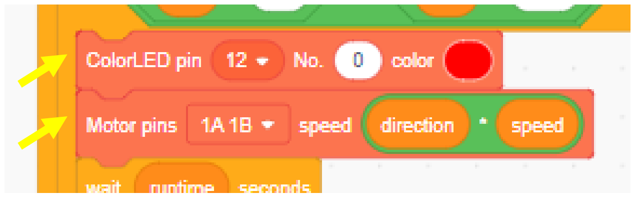
This is where changes can be made in motor speed, motor direction, and/or run time, defined specifically in the three white oval areas (see yellow arrows).

Next, notice the variables below as to how the microphone is to function; in this case, a range from >300 to <1023. This is potentially where one could change the sound sensitivity of the microphone. We will not be making changes here.



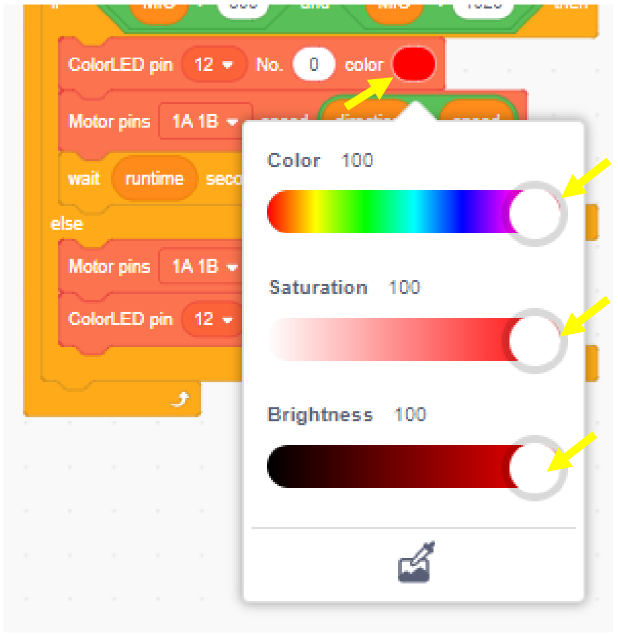
```
forever
  set MIC to Analog read pin A6
  if MIC > 300 and MIC < 1023 then
    ColorLED pin 12 No. 0 color
```

Next, notice both the ColorLED and the Motor blocks below;



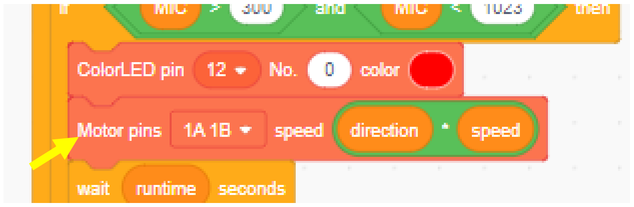
```
ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
wait runtime seconds
```

You can see that the ColorLED pin is assigned to #12 on the project hub. That will remain the same. However, notice the red oval. This is where one could change the LED color. Watch below what happens when one clicks on the red oval;

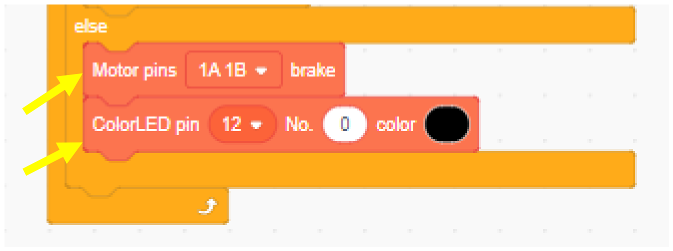


By clicking on each white button and sliding back them back and forth, one can achieve a desired color. Notice how the red is achieved above, (in this instance, all buttons are vertical to each other.)

Next note the location of the Motor pins 1A1B block below. If a second block of this type is added below the first one, then a second motor could be added to the project.


MIC > 300 and MIC < 1023 then
ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
wait runtime seconds

Finally, the Scratch program/script ends with braking or stopping the motor and then turning off the ColorLED, (black oval.)



So far, you may already have guessed that one can have a fascinating time changing any number of variables. Be mindful however that each time you change a variable and want to see its effect on your project, you will need to "Flash Firmware" (in Arduino mode) to upload the program/script to your Windmill CPU/project.

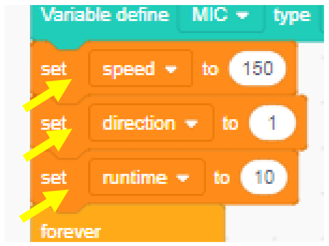
Let's Have Some Fun!

Let's try the following;

- The Windmill has a slower speed
- The Windmill turns in the opposite direction
- The Windmill stays on a shorter time
- The Windmill changes LED color
- The Windmill has two motors

Change Windmill Speed

Remember previously, (on page 4), we showed you three variable blocks of speed, direction and runtime.

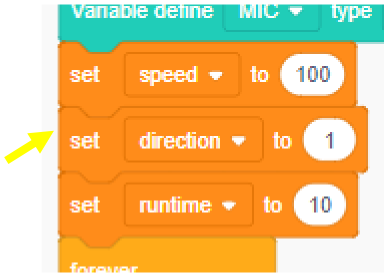


```
Variable define  MIC  type
set  speed  to  150
set  direction  to  1
set  runtime  to  10
forever
```

If, we click on the speed value of 150 on set speed, and type in the value of 100 in its place, what do you suppose will happen? If you answered that the motor will turn slower, your answer is correct. Go ahead, change the value to 100 (maximum is 255), upload the altered script into the Windmill and see what happens.

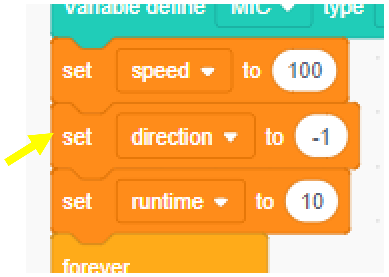
Change Direction

Notice the set direction block below.



```
Variable define  MIC  type
set  speed  to  100
set  direction  to  1
set  runtime  to  10
forever
```

Inserting a negative value (-1) reverses the motor rotation direction.

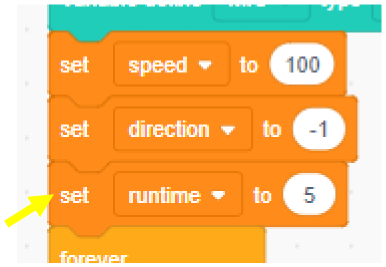


```
Variable define  MIC  type
set  speed  to  100
set  direction  to  -1
set  runtime  to  10
forever
```

Go ahead, change the direction value to -1 in your script, upload the altered script into the Windmill and see what happens. Having fun?

Change Run Time

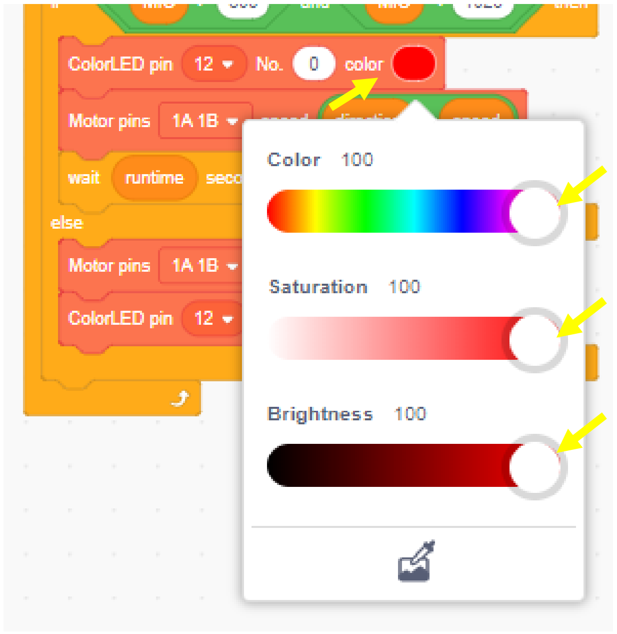
Change the motor run time by changing the set runtime value below. In this case, insert the number 5 (for 5 seconds).



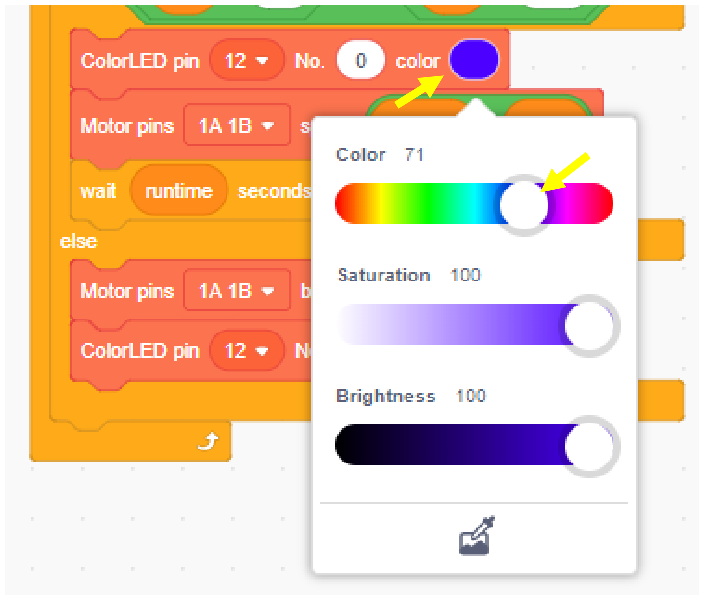
```
set  speed  to  100
set  direction  to  -1
set  runtime  to  5
forever
```

Your project will run for 5 seconds instead of 10. Go ahead and make that change now, upload it and see what happens.

Change LED Colors



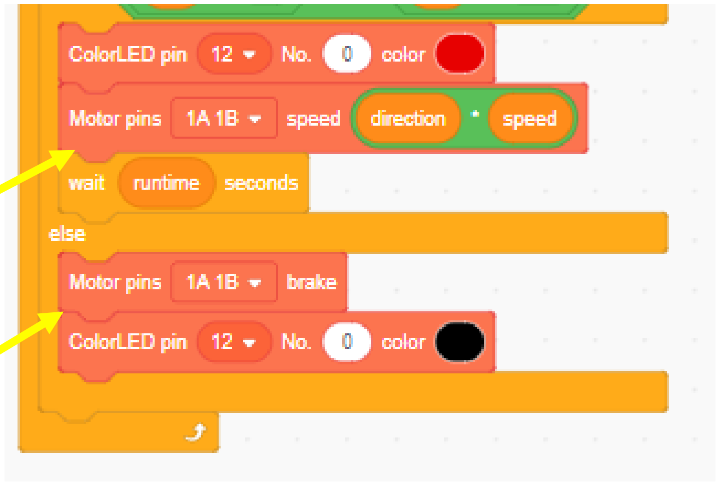
Earlier, we pointed out how to gain access to color changes by clicking on the red oval above in the ColorLED block. By clicking on any of the white buttons above, and sliding them back and forth, (with your cursor) any number of different colors can be achieved by varying the color, saturation and brightness. For our example, let's change our ColorLED to blue. Go ahead and put your cursor over the top white button (see next column) click and slide it to the left over the blue area of the color spectrum. Notice how the red color in the ColorLED block oval now turns to blue as illustrated in the right column.



Go ahead, change the color in your script as indicated, upload the altered script into the Windmill and see what happens. Blue, right? Good job.

Add an Additional Motor

Look at the script below and notice one block is dedicated to one motor only along with direction and speed. Also, it only follows that once a motor is started, it must be stopped or braked. See the Motor brake block below as well.

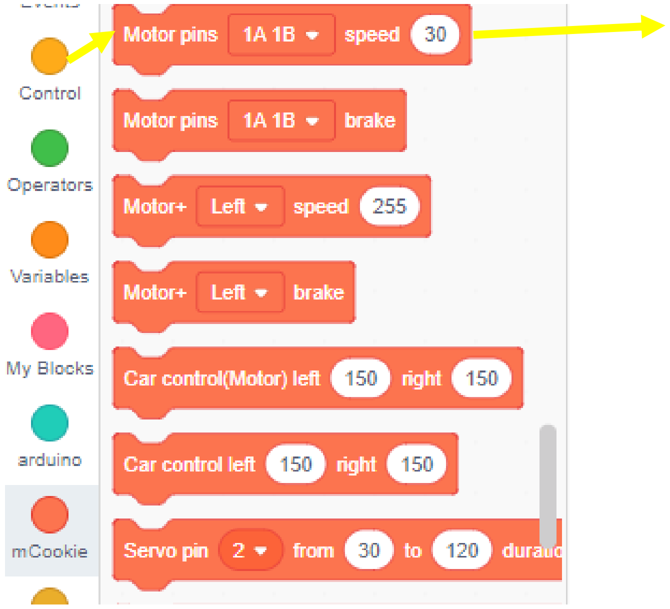

```
ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
wait runtime seconds
else
Motor pins 1A 1B brake
ColorLED pin 12 No. 0 color
```

To add a second motor then to our project, a new Motor block must be inserted into the existing script above, (see top yellow arrow insert point), plus we need to add a second Motor brake block as well, (see bottom arrow insert point).

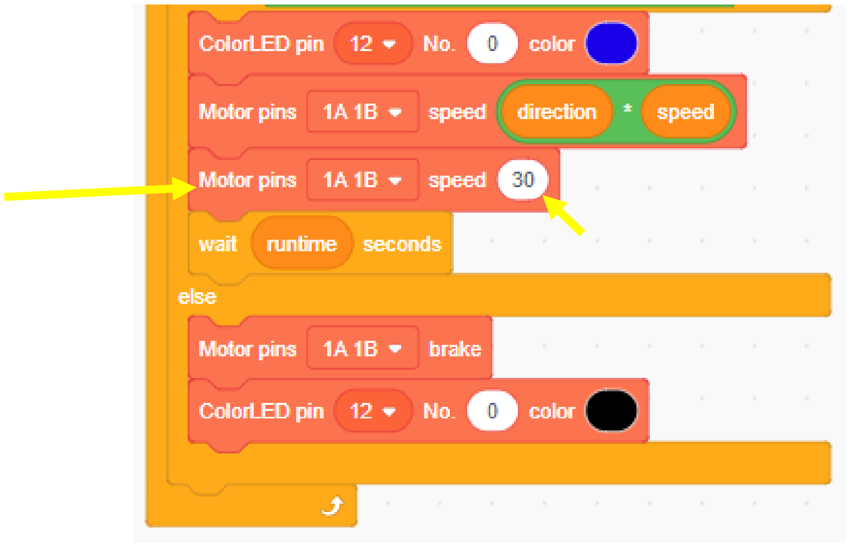
Let's begin by adding the second Motor block. Start by clicking on the Motor block from the mCookie group on the left pane of mDesigner. (see next column.)

Drag the Motor block and insert it into place below the first Motor block in our script. (see next column).

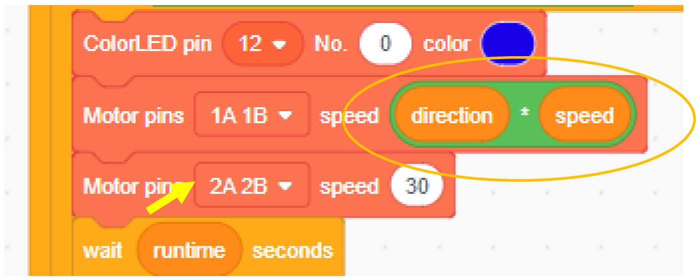
Notice how this new block says speed 30. (We will need to change this value in a few minutes.)


Control
Operators
Variables
My Blocks
arduino
mCookie
Motor pins 1A 1B speed 30
Motor pins 1A 1B brake
Motor+ Left speed 255
Motor+ Left brake
Car control(Motor) left 150 right 150
Car control left 150 right 150
Servo pin 2 from 30 to 120

Your script should now look like this;

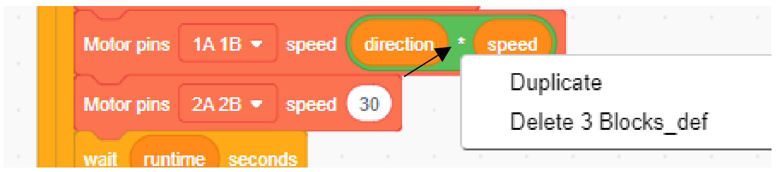

```
ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
Motor pins 1A 1B speed 30
wait runtime seconds
else
Motor pins 1A 1B brake
ColorLED pin 12 No. 0 color
```

Next, change the new Motor block to pin 2A2B, (see arrow).

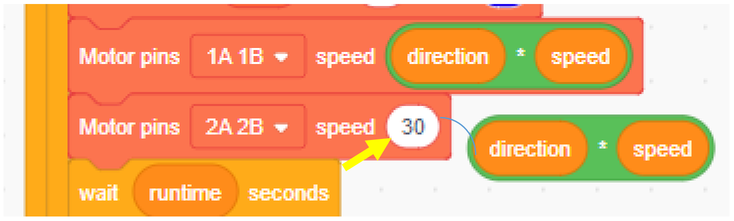

ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
Motor pins 2A 2B speed 30
wait runtime seconds

Notice from above (yellow oval) we need to somehow replicate the direction and speed variables in the top Motor block and insert them in the bottom Motor block.

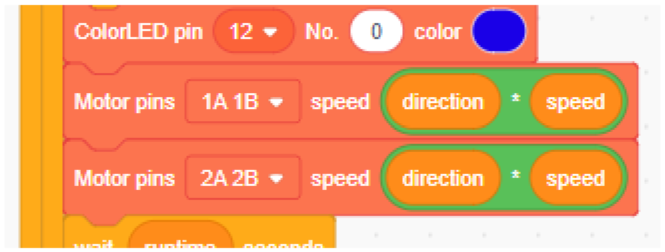
First place your cursor over the times symbol *, (see arrow below) right click, select Duplicate, and the direction and speed block can be replicated as one unit…


Motor pins 1A 1B speed direction * speed
Motor pins 2A 2B speed 30
wait runtime seconds
Duplicate
Delete 3 Blocks_def

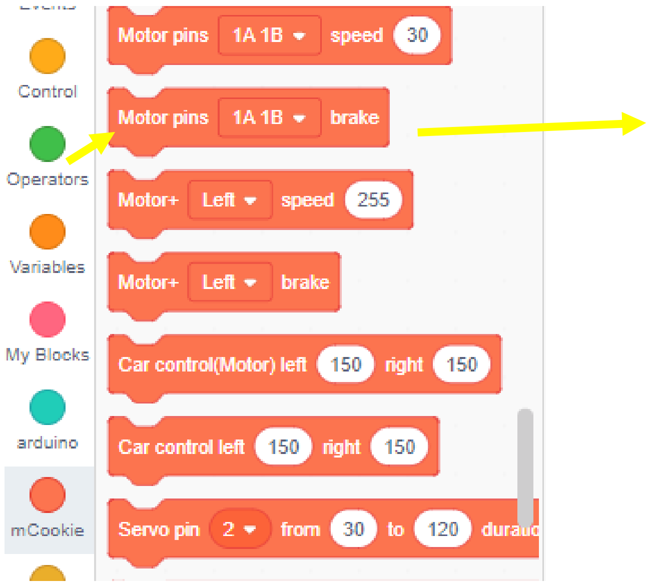
and then dragged over and into the speed 30 oval below.


Motor pins 1A 1B speed direction * speed
Motor pins 2A 2B speed 30
direction * speed
wait runtime seconds

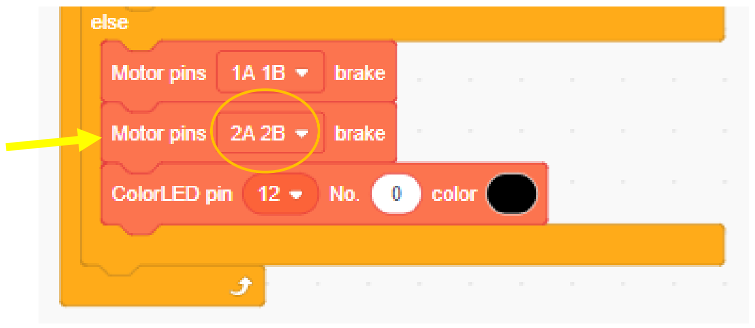
Your script should look this now…


ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
Motor pins 2A 2B speed direction * speed

The next and final step is to add and insert the second Motor brake into our script. Notice the Motor brake block location in the mCookie section of mDesigner.


Control
Operators
Variables
My Blocks
arduino
mCookie
Motor pins 1A 1B speed 30
Motor pins 1A 1B brake
Motor+ Left speed 255
Motor+ Left brake
Car control(Motor) left 150 right 150
Car control left 150 right 150
Servo pin 2 from 30 to 120

Click and drag this block over to the script area, and then insert the second Motor brake block between the first Motor brake block and the ColorLED block in the script area. Don't forget to change the pin number to 2A2B (see circle). Your script should look like the following;

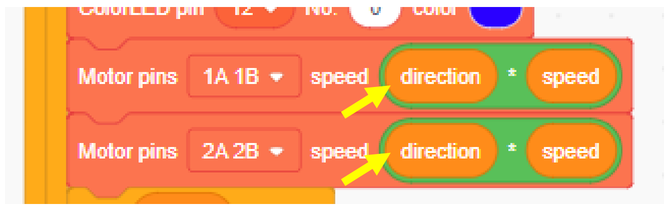


This now completes our new altered script. If you haven't already, go ahead make changes in your own script, and save it with a new file name. Then upload it into your Windmill project and see what happens. Both motors should be working now... but in the same direction.

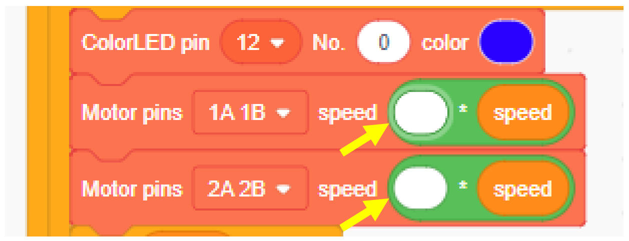
Bonus - Motors Running in the Opposite Direction

For some extra fun, you can change your Windmill script so that the two motors rotate in opposite directions. Here's how in two easy steps.

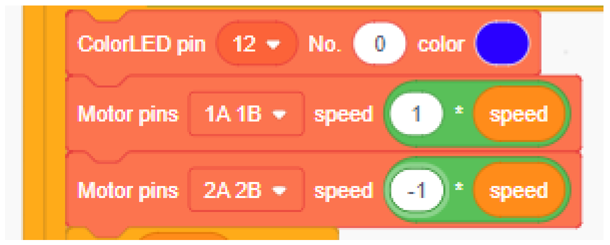
Notice the "direction" ovals in the next column. We need to remove those and insert the value of 1 in the top oval and then -1 in the second.


ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed direction * speed
Motor pins 2A 2B speed direction * speed

1. First click on one "direction" oval at a time and drag them over to the left palette area, (they should disappear). Your script should now look like the following;

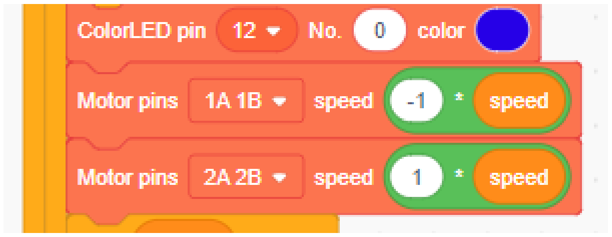

ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed * speed
Motor pins 2A 2B speed * speed

2. Enter in a value of 1 in the top Motor block and -1 in the bottom Motor block. (See below.)


ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed 1 * speed
Motor pins 2A 2B speed -1 * speed

This now completes our new altered script. If you haven't already, go ahead make the final changes in your own script, and save it with a new file name. Then upload it into your

Windmill project and see what happens. Both motors should be working now… but in the opposite direction! If you want to reverse directions the opposite way, place a -1 in the top and a 1 on the bottom oval. See below.


ColorLED pin 12 No. 0 color
Motor pins 1A 1B speed -1 * speed
Motor pins 2A 2B speed 1 * speed

Finally, and if necessary, don't forget to re-save your entire new Windmill (opposite direction) script files you have just created.

Congratulations!

You should now have a Windmill project that now does the following;

- The Windmill has a slower speed
- The Windmill turns in the opposite direction
- The Windmill stays on a shorter time
- The Windmill changed the LED color (blue)
- The Windmill has two motors that turn in opposite directions.

Changing project variables, as you now have already seen, is not as difficult as you may have thought. You may be very pleased to learn that changing the way an Itty Bitty City project operates is not so difficult, and just doing this can give you hours of fun and experimentation! Remember, with Itty Bitty City there are no bad mistakes, just learning experiences.

Troubleshooting

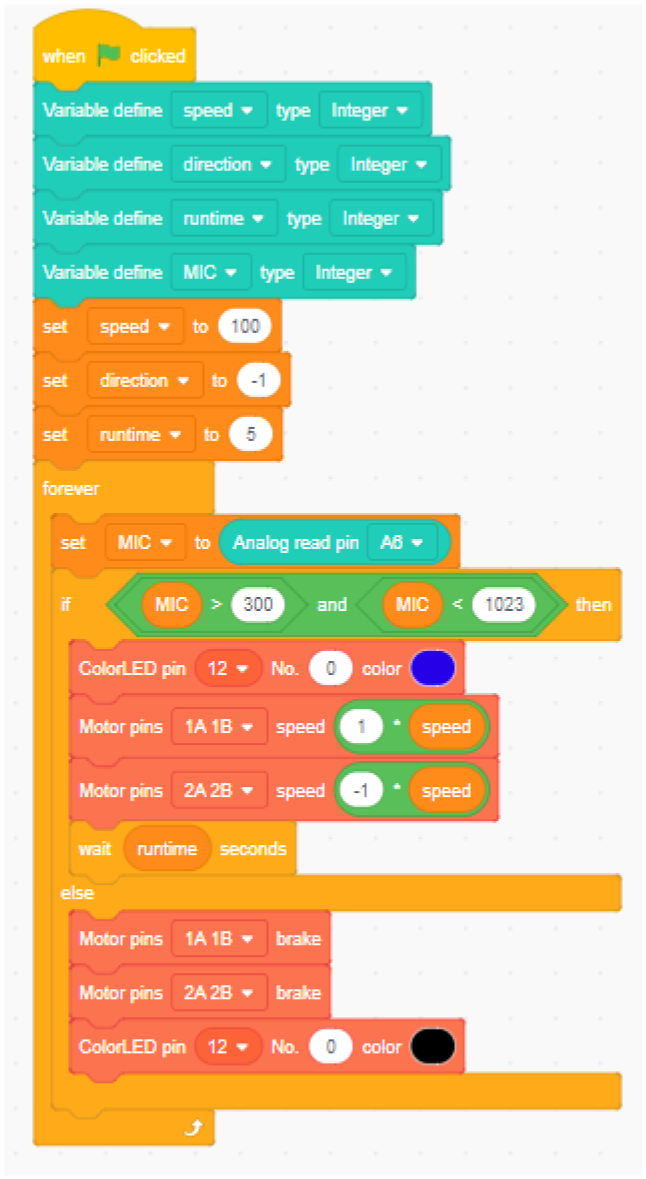
If your project is not working correctly, *do not panic*! Here are some trouble-shooting hints:

- Is your PC and Windmill connected?
- Is the battery in the Windmill have the red LED on?
- Have you selected the correct serial port?
- Are you in the Arduino mode?
- Did you click on Flash Firmware?

Finally….

For your convenience and as a point of reference, we have included a new altered Windmill script on the next page;

Modified Windmill Script



```
when 🏁 clicked
Variable define speed type Integer
Variable define direction type Integer
Variable define runtime type Integer
Variable define MIC type Integer
set speed to 100
set direction to -1
set runtime to 5
forever
    set MIC to Analog read pin A6
    if MIC > 300 and MIC < 1023 then
        ColorLED pin 12 No. 0 color ●
        Motor pins 1A 1B speed 1 * speed
        Motor pins 2A 2B speed -1 * speed
        wait runtime seconds
    else
        Motor pins 1A 1B brake
        Motor pins 2A 2B brake
        ColorLED pin 12 No. 0 color ●
```