**Using Snap! with MicroBlocks to Control Microcontrollers**

The first higher level computing language, FORTRAN (Formula Translation), was designed for processing numbers. The second higher level language in use today, LISP (List Processing), was developed by Joseph McCarthy in the M.I.T Artificial Intelligence (AI) Laboratory. In the 1960s the co-director of the MIT A.I. Lab, Seymour Papert, developed Logo. Logo was the first computing language developed for use by children. (The name 'Logo" is not an acronym; it is the Greek word for "Word.")

Alan Kay and a team of researchers at the Xerox Palo Alto Research Center (PARC) developed many of the modern computing concepts in common use today, including a graphical user interface with windows and a mouse. Alan Kay designed a computing language, Smalltalk, designed for use by children that was influenced by Logo, among other languages. Smalltalk introduced object-oriented programing and messaging, concepts that are now incorporated into modern computing languages from Java to Python.

*Figure 1. Dan Ingalls Demonstrates Smalltalk (Source: Computer History Museum)*

Alan Kay and his colleagues demonstrated Smalltalk for Steve Jobs, who incorporated many of the ideas about its interface into the Macintosh computer. Smalltalk also influenced the development of Scratch, Snap*!*, and MicroBlocks (Figure 2).

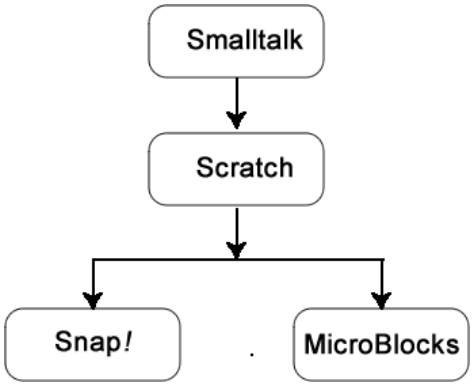



*Figure 2. Overview of Developmental Relationship between Snap! and MicroBlocks*

Smalltalk was initially developed in the 1970s. Working with Alan Kay near the end of the twentieth century, John Maloney helped create a new implementation of Smalltalk called Squeak. He then used that to create Etoys, a blocks-based language for children. Jens Moenig also had the opportunity to work with

Smalltalk on several different occasions and also worked with Alan Kay. These experiences with Smalltalk would prove to be crucial in subsequent development of both Scratch and Snap!

In 2002, John joined the Lifelong Kindergarten group at the MIT Media laboratory and began work on Scratch, building on ideas from Etoys. The first version of Scratch was written in Smalltalk. Scratch has become the most widely used children's language today, and it is currently the 22nd most popular computer language overall according to the TIOBE index (https://www.tiobe.com/tiobe-index/).

The initial version of Scratch did not provide users with the ability to create new blocks. Jens Moenig developed a version of Scratch, Build Your Own Blocks (BYOB), that included this capability. Dan Garcia, a professor of computer science at the University of California, Berkeley, asked Jens Moenig if he could use BYOB in a new computing course, the *Beauty and Joy of Computing* (BJC), developed at Berkeley. Parents objected to use of BYOB as the name of a computing language used by students. Another Berkeley professor, Brian Harvey, collaborated with Jens to develop Snap*!*, a version of BYOB that included advanced concepts suitable for undergraduate computing courses. Today Snap*!* and the Beauty and Joy of Computing are not only used at Berkeley, but by more than a thousand high school computer science teachers.

In recent years small, inexpensive microcomputers known as microcontrollers have transformed modern engineering. A microcontroller, as its name suggests, can be used to control physical objects through use of motors and actuators. In engineering these devices are designed to be embedded in mechanisms that they control. A microcontroller connected to a laptop can extend the capability of the laptop to control physical objects.

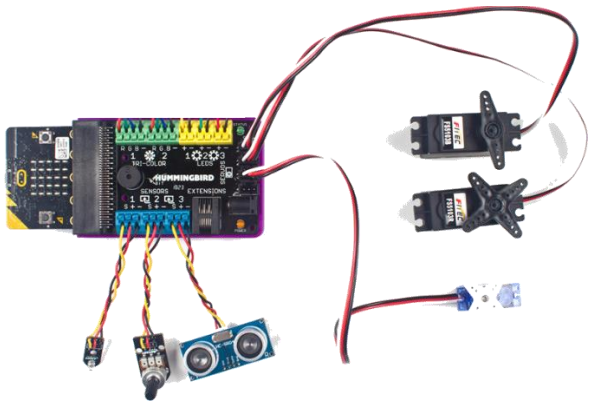



*Figure 3. A Microcontroller with Sensors and Actuators*

Snap*!* can be run from a browser and can be accessed from Windows, Macintosh, and Linux operating systems. However, Snap*!*, by itself, does not speak the language of microcontrollers. John Maloney developed a language, MicroBlocks, designed specifically for microcontrollers. MicroBlocks, like Snap*!*, is influenced by Smalltalk and Scratch. MicroBlocks makes it possible to write code for a microcontroller using a laptop.

MicroBlocks consists of a development environment that runs on a laptop and a virtual machine (VM) that runs on the microcontroller. The development environment on the laptop translates blocks code into instructions that are executed by the VM to make the microcontroller do things. As you work, the instructions for your scripts are sent over the USB (Universal Serial Bus) cable to the microcontroller and stored in memory, ready to be run on a moment's notice. This allows you to test your code frequently as you work without waiting for the entire program to compile and download. Furthermore, your code is

stored in persistent Flash memory, so you can disconnect the microcontroller from the computer, plug in a battery, and your program will start right up. Since microcontrollers are tiny and don't need much power, you can create projects that you can carry in your pocket, mount on a skateboard, or even wear.

There are instances in which it may be useful to use the microcontroller as an extension of the laptop. For example, a videogame could be developed on the laptop that uses accelerometers and other sensors on the microcontroller as inputs to control the game. In that case, it would be useful to take advantage of the computational capabilities of Snap*!* on the laptop while accessing the physical computing capabilities of MicroBlocks on the microcontroller. The steps involved in this process are as follows:

1. Connect MicroBlocks to the microcontroller and create a program.
2. Disconnect Microblocks from the microcontroller.
3. Connect Snap*!* to the microcontroller.
4. Create a program in Snap*!* that interacts with the program on the microcontroller previously developed in Step 1 above.

The crucial thing to understand about this process is that only one program at a time can communicate with the microcontroller via the serial port (i.e., through the USB cable connecting the microcontroller to the laptop).

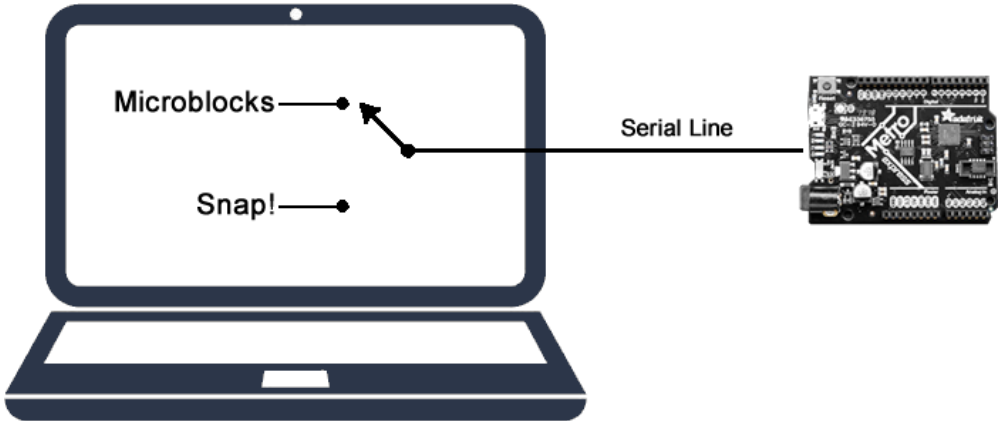



*Figure 4. Connecting MicroBlocks to the Microcontroller*

When the microcontroller is connected to the laptop via a USB cable, the MicroBlocks program on the laptop can be connected to the microcontroller by clicking the icon of the USB plug.

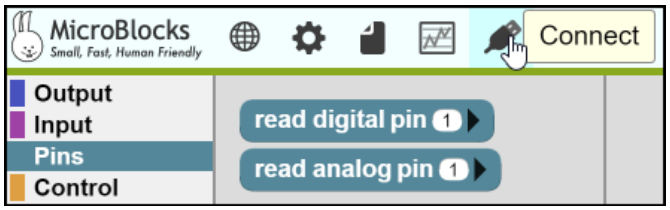



*Figure 5. Selecting the USB Icon in MicroBlocks*

This will produce a dialog box indicating that MicroBlocks wants to connect to the microcontroller (in this illustration, an Adafruit Metro M0 Express).

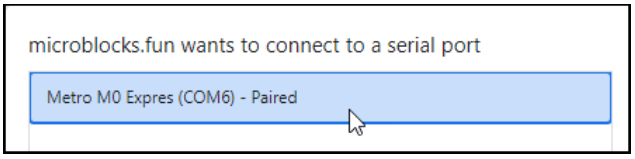



*Figure 6. Selecting a Microcontroller*

Once a connection is made, a green circle will appear around the USB icon.


*Figure 7. A green circle around the USB icon indicates that a connection has been made*

The MicroBlocks program says that when the microcontroller receives the message, "Pin 3 On" from Snap!, it should set digital pin 3 on the microcontroller to "On."


*Figure 8. Creating a MicroBlocks Program*

This program can be executed by clicking the green Start button in the upper right-hand corner of the MicroBlocks window.

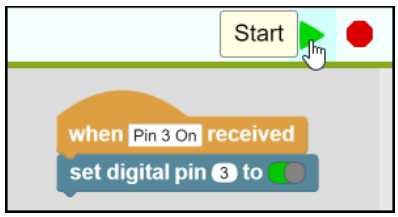
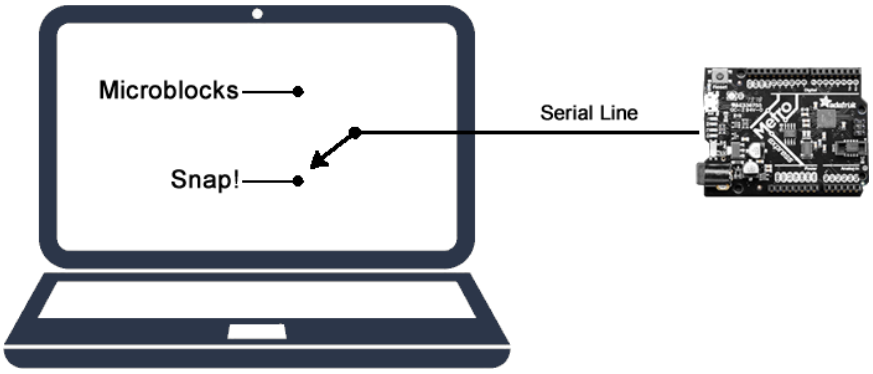



*Figure 9. Running the MicroBlocks Program*

Once MicroBlocks has been used to create a program on the microcontroller, MicroBlocks can be disconnected from the microcontroller by clicking the USB icon again. A dialog box will appear with the option to disconnect. Once MicroBlocks has been disconnected from the microcontroller, the green circle should no longer be present around the USB icon. However, the MicroBlocks program will still be present on the microcontroller even after it is disconnected from the serial port.


*Figure 10. Disconnecting MicroBlocks from the Microcontroller*

Once MicroBlocks is disconnected, Snap*!* can be connected to the microcontroller.




*. Figure 11. Connecting Snap! to the Microcontroller*

Snap*!* can be connected to MicroBlocks by loading a MicroBlocks library that adds a new MicroBlocks palette to Snap*!*, extending its capabilities.

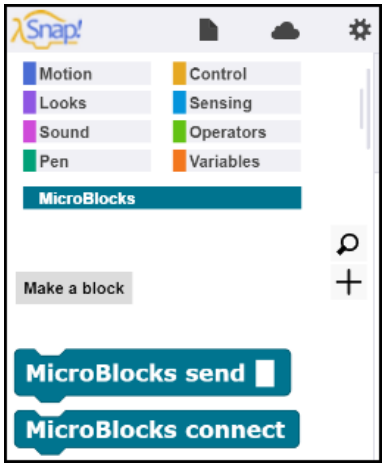



*Figure 12. The MicroBlocks Palette in Snap!*

The **MicroBlocks Connect** block can be used to connect Snap*!* to the microcontroller.


This block will produce a dialog box indicating that "Snap.berkeley.edu wants to connect to a serial port."

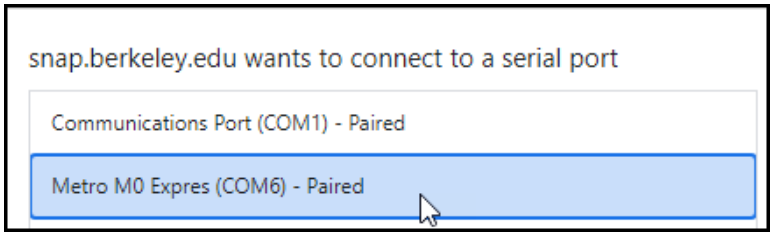



*Figure 13. Selecting a Microcontroller*

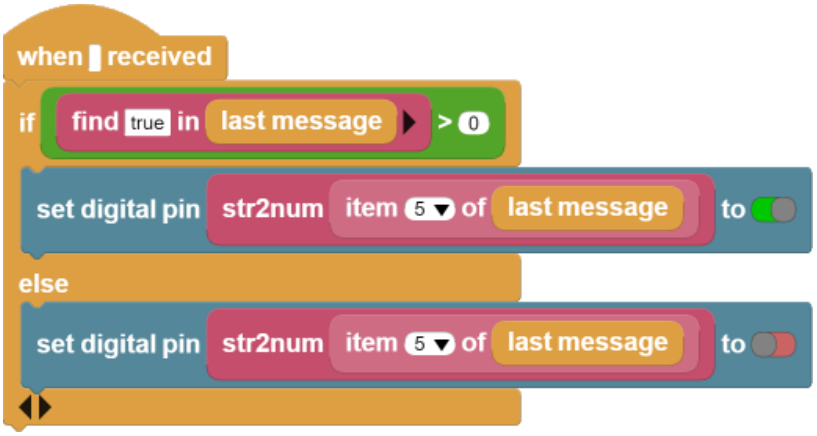
Once Snap*!* is connected to the microcontroller, the **MicroBlocks Send** block can be used to send a message to the MicroBlocks program on the microcontroller.


The message, "Pin 3 On", is based on the assumption that there is corresponding code on the microcontroller waiting to receive this message. (Note: If MicroBlocks is still connected to the microcontroller, the Snap*!* **MicroBlocks Send** block will generate an error message.)

A Snap*!* output block can be created to generate the messages to the microcontroller.


This block can be used as an input to the Snap*!* **MicroBlocks Send** block.

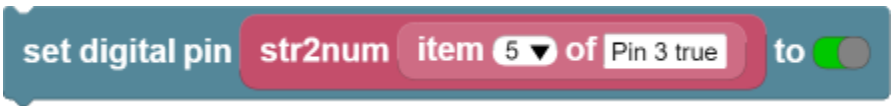

This block will generate the text string "Pin 5 true" that is sent to the microcontroller. A corresponding MicroBlocks program must be available on the microcontroller to interpret this string. The **Last Message** block contains the text string in the last message sent. In the example below, the MicroBlocks program checks to see if the string "true" is present in the message.




If the string "true" is present in the message, the MicroBlocks program can identify the pin number because it is the fifth character in the string.

This character must be converted to a number by using the StringToNumber block:


Once this is done, the output can be used to turn on the specified digital pin.




Although Snap*!* does not natively speak the language of microcontrollers, the combination of Snap! and MicroBlocks together can be used to enable Snap*!* to interact with microcontrollers to control physical objects. One of the most useful features of MicroBlocks is that it permits the same code to be used on many different microcontrollers. In this case, MicroBlocks serves as an interpreter that translates a block such as **Set Digital Pin 2 to True** into the native language of different microcontrollers.

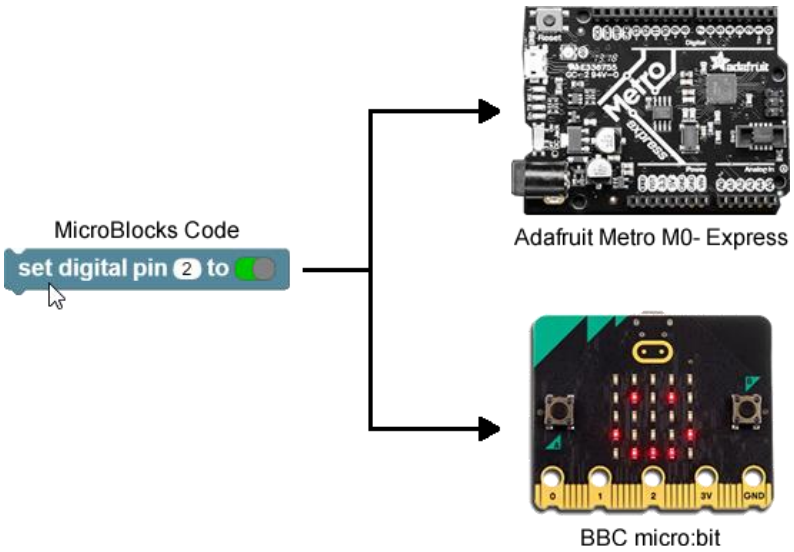



*Figure 14. MicroBlocks Translates Code into the Language of Different Microcontrollers*

New generations of microcontroller are constantly being invented, often with additional features at lower cost. By using MicroBlocks as an intermediary to develop code for microcontrollers, the software routines developed for one family of microcontrollers can be used with more powerful microcontrollers that may emerge in the future.

Snap! and MicroBlocks both have their roots in prior generations of languages like Lisp and Smalltalk. This common heritage means that they are based on similar philosophical foundations, and work well together.