Module 4, Lesson 1 – Wireless sensor networks

? minutes

Purpose of this lesson

- Learn about remote sensing, distributed sensor networks, and issues of wireless data transmission
- Recap the use of “My Blocks” to simplify and clarify programs
- Recap on data-logging with the NXT

Materials

Copy of the lesson

Computer with Mindstorms software

1 NXT

1 sensor (e.g. the temperature sensor from modules 1 and 2)

Introduction - distributed wireless sensor networks

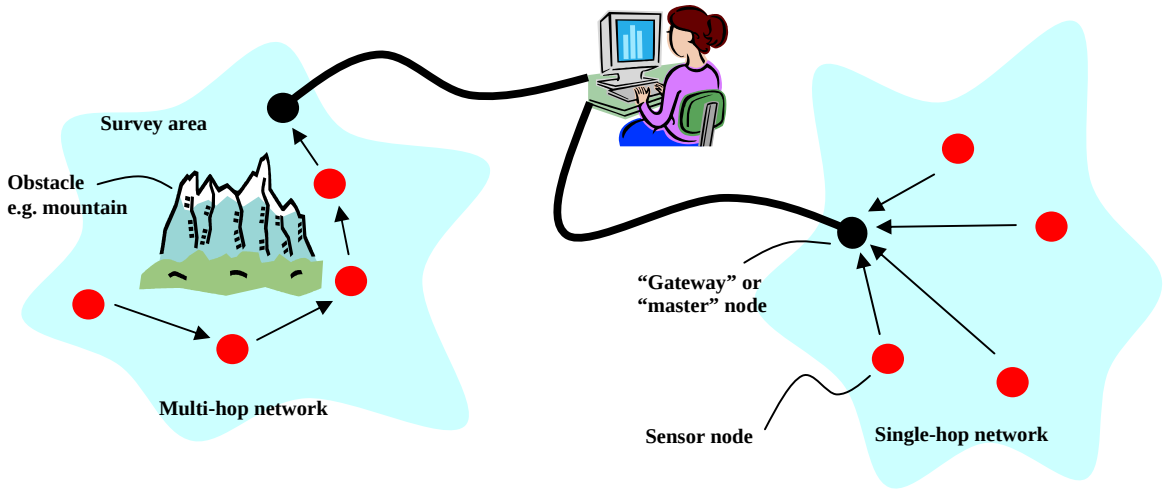


Figure 1. In a “single hop” network (right), each sensor node transmits its data directly back to a gateway node, from which it can be delivered to the scientist’s computer for analysis. If the data has to be transmitted over very large distances, or around an obstacle, then it may be necessary to use a “multi hop” network (left) in which data is relayed from one node to the next along a chain of communicating wav-stations.

A *wireless sensor network* (WSN) consists of a group of sensor "nodes" which are scattered over a region of interest. Together, this group of sensors is used cooperatively to measure and monitor the region. In environmental applications, water temperature or pollutants might be measured by the sensors at many nodes in different locations across the region, to build up a map of how these quantities vary over the region. This is sometimes known as "area monitoring". In military applications, sensors might be scattered across a battlefield to detect and monitor the movements of enemy vehicles. In industrial applications, sensors might be positioned in many different places across a large factory, to ensure that the factory processes are all running correctly.

Such sensors could communicate their information through cables or wires. However, laying cables is expensive and time consuming and this expense and difficulty increases as the distances between sensors increase - a few inches of wire is cheap, but a few miles of wire is very expensive indeed! For this reason, it is useful to build sensor networks without any wires connecting the sensors – i.e. "wireless" sensor networks. Wireless Sensor Networks (WSNs) use radio transmitters to send their information back to a scientist at a central monitoring location. This means that sensors can be installed without fixed wires, and so it is easy to add additional sensors to the network, or to move an old sensor to a new location, without changing the underlying infrastructure of the system.

In module 4 of the SENSE-IT program, you will build and program your own wireless sensor network. Your network will have three remote nodes. At each node, an NXT computer will record multiple sensor "modalities" including temperature, turbidity, salinity and depth, using the sensors which you created in earlier modules of the project. Each of these remote modes will then wirelessly transmit its data back to a master or "gateway" node (another NXT computer) which will perform continuous data-logging over a period of time. Then you will be able to upload the data files to your PC, where you will be able to view and analyze all four environmental parameters, from all three locations, and see how these have varied over time.

Each sensor node will typically include the following important elements:

- One or more sensors of various kinds (e.g. temperature, salinity and turbidity for water quality measurements).
- A power source (e.g. batteries, solar panels, wind turbines).
- A radio transmitter to report the data measured by the sensors.
- A small computer to collect data from the sensors and create data files for transmission by the radio transmitter.

Discuss the following engineering design issues with your classmates and your teacher:

- *Energy source*: how long can a sensor be deployed without human intervention or servicing? How much power do the sensors and transmitters require? How long will a battery last? Is an extra source of power required, like solar panels or a windmill, to top up the batteries so that they do not need regular replacement? Sending an engineer to visit a sensor to replace its battery costs time and money. What are the tradeoffs between this and buying longer lasting (more costly) batteries, or building expensive solar panel systems to recharge the batteries?

- *Transmitter:* how far does each sensor node need to transmit its data? Should you use a large expensive transmitter to cover this distance in a single "hop" (see figure 1)? Or could

each node user a smaller and cheaper transmitter to send data over a shorter distance to the next node, and thereby send the data back to the scientist in a series of hops ("multi hop" network). How much energy do these transmitters require? Do your sensor nodes have enough battery power to power the long range transmitters, or would using a series of short range transmitters allow the sensors to last longer without having their batteries replaced?

- *Sensor locations:* where should we position our sensors, in order to build a meaningful and representative map of what is going on in our region of interest? Should we concentrate our sensors in the most variable or interesting parts of the region – if so, do we risk missing an event that happens elsewhere? How many sensors do we need, and how densely should we space them? If water salinity varies a lot over small distances then we must place our salinity sensors close together if we want to observe this variation-otherwise we cannot measure it. Should we use a small number of accurate (and expensive sensors) or a large number of cheap (and less accurate) sensors?

Note: there are not necessarily any right or wrong answers to these questions. In each particular sensor problem, the engineer will have to weigh up the pros and cons of different approaches and arrive at the best compromise design – this may also be known as a problem of "design trade-offs". Different problems in different contexts will require different solutions.

What you will do in the rest of this module:

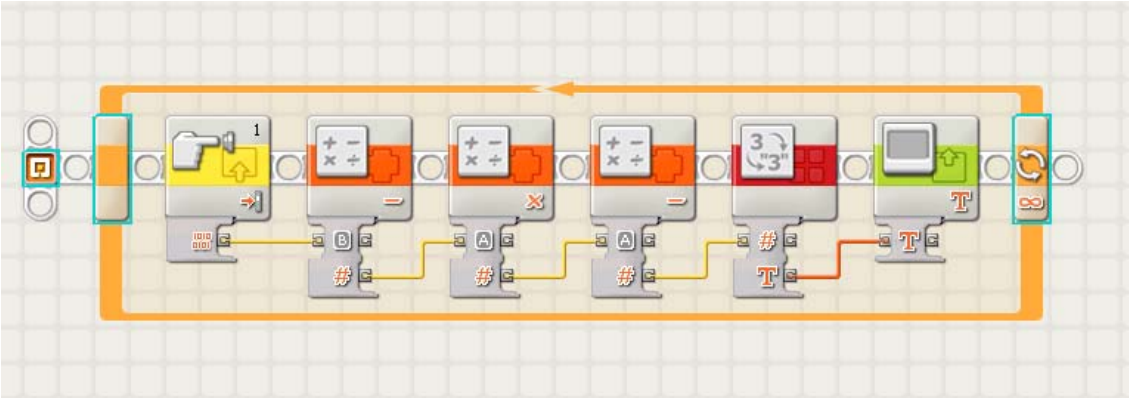
In the next two lessons, you will actually build an entire wireless distributed sensor network in your classroom. This will require additional NXT programming. Therefore, in the remainder of this lesson, we will review a few aspects of NXT programming, including the use of "My Blocks" and also data-logging. By the end of this lesson, you will have built one complete "sensor node", i.e. an NXT connected to four different sensors (temperature, salinity, turbidity, depth), which you have programmed to record data continuously from all of the sensors and save it as a data-log in a text file, which you can then upload onto your PC and analyze. In later lessons you will learn how to connect many of these nodes together in a sensor *network*, by making them transmit their data wirelessly.

Part 1 – a recap on using My Blocks feature

1) Go to the Help menu in your Mindstorms programming software. Read the help section on "My Blocks" – a copy is also attached at the end of this lesson.

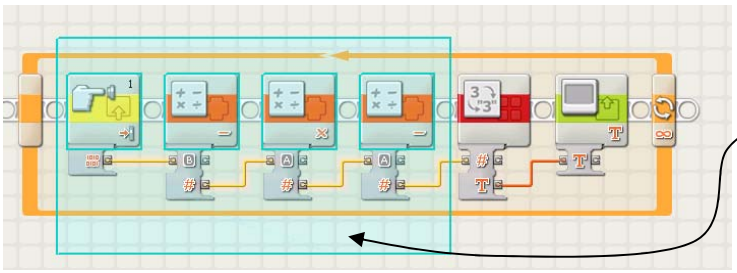
In module 4, your NXT programs will become longer and more complicated than in previous modules, and you need to find ways of simplifying and clarifying them. One way to do this, is to take a section of old program (a section of your "code") and save it as a single "My Block" which can then be re-used in new programs.

2) Find the program you wrote for displaying readings from your temperature sensor in module. It should look something like this:

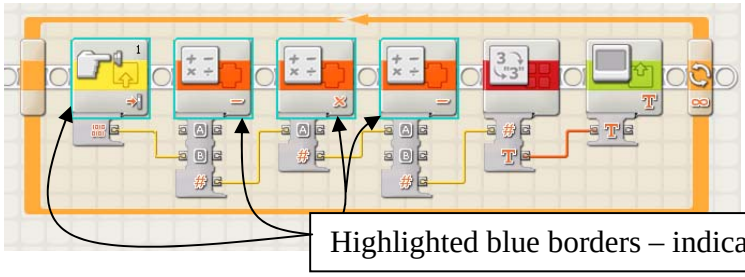


NOTE: This program is only an example. Your own particular program might look different. For example you might have four math blocks instead of the three shown here-there are many different ways to write a working program for this sensor and it is good to encourage students to be creative and try out slightly different ideas to each other in their programs. The important points are:
i) make sure that whatever code you have actually works before you turn it into a MyBlock!
ii) make sure that *you* select the relevant "core" part of *your* code for turning into a MyBlock (see next step…)

3) Look at the "core" part of this code (the part which takes values from the sensor and converts them into temperature °C). Highlight this section:



Note: try clicking outside the orange repeat loop before dragging the blue highlighted area over the program blocks (as in the picture) – sometimes the software doesn't let you click inside the orange loop. Alternatively, hold down the shift key while clicking on each block that you wish to select.

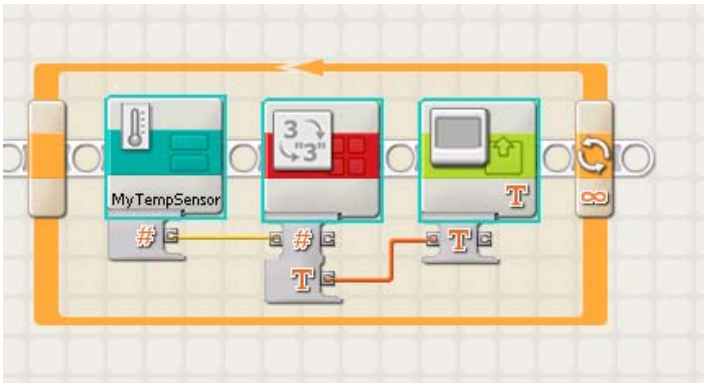


Highlighted blue borders – indicates which blocks you selected

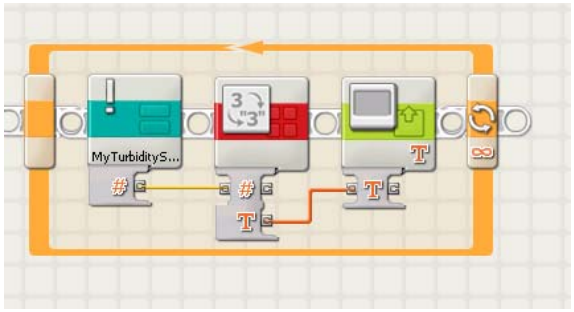
Note: the "drag area" will only remain shaded blue, as shown in above picture, until you release the mouse button. After releasing the mouse button, the blue shaded area will disappear, and each block that you selected will retain a blue border to let you know that you successfully selected it.

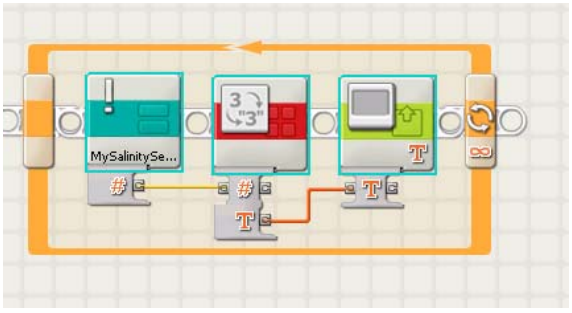
Note: do not highlight the whole loop – only the "core" part of the code which takes a sensor value and converts it into °C, i.e. don't include the blocks for displaying on screen.

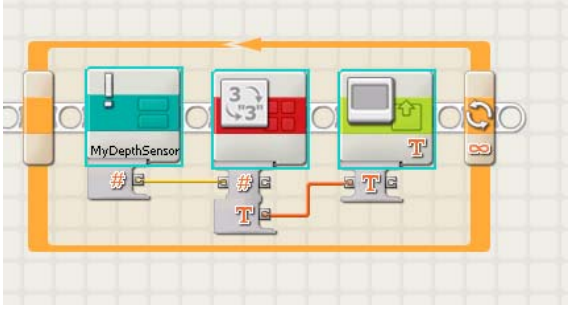
4) Now click on the “Edit” menu and select “Make A New My Block” to turn the highlighted section of code into your own temperature measurement block. Now your program for sensing temperature, converting to °C, and displaying on the screen looks much more simple and concise:



5) Now repeat this exercise for the salinity, turbidity, and depth sensor code:







Important!

Note: the “My Block” for each sensor will only read from whichever sensor port was designated in the original code for that sensor.

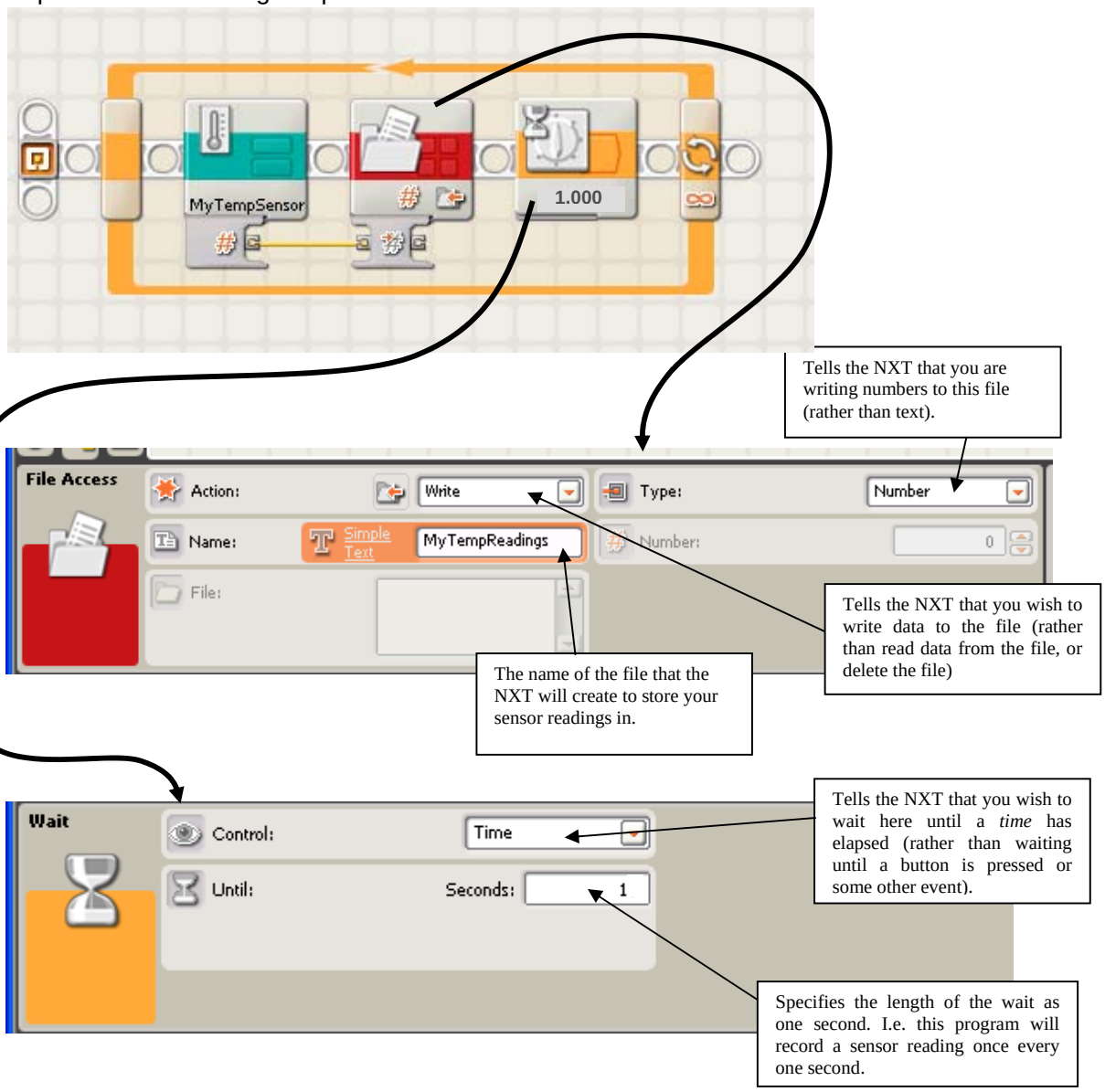
If you want to move the sensor to a different port (e.g. when using multiple sensors), then you must open up the My Block and edit the code inside it to make it read from the correct sensor port.

To open up a My Block and edit the code inside it, simply double click on it. This is also explained in the Mindstorms help file.

Part 2 – recap on data-logging with a single NXT:

You may have encountered data-logging previously when working on module 2 of this project. In part 2 of this lesson, we will review the data-logging material by combining it with the My Blocks exercise from Part 1.

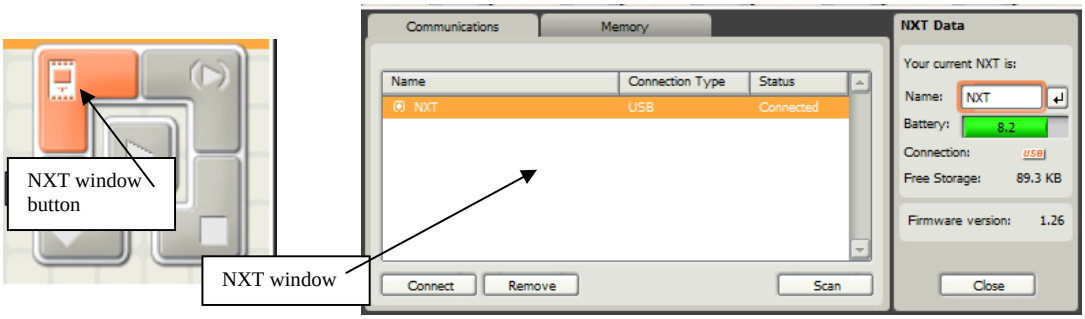
1) Create a program to data-log readings from a thermistor, using the My Block that you created in part 1 for measuring temperature:

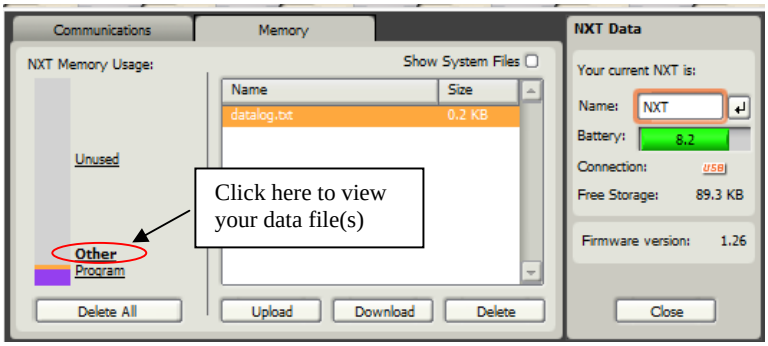


Now run this program. After about 20 seconds, stop the program.

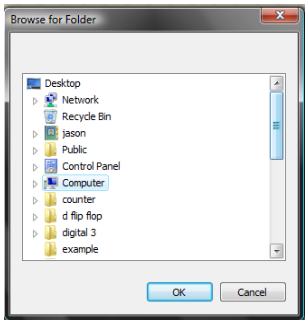
Once the program has stopped, connect your NXT to the PC via the USB cable, and upload the file “MyTempReadings.txt” to your PC, then open the file and check that the numbers seem sensible.

To upload your data file from the NXT to the PC, first click on the NXT window button, shown below, to bring up the NXT window. After scanning for a few seconds, it should show that your NXT is connected under the Status column.





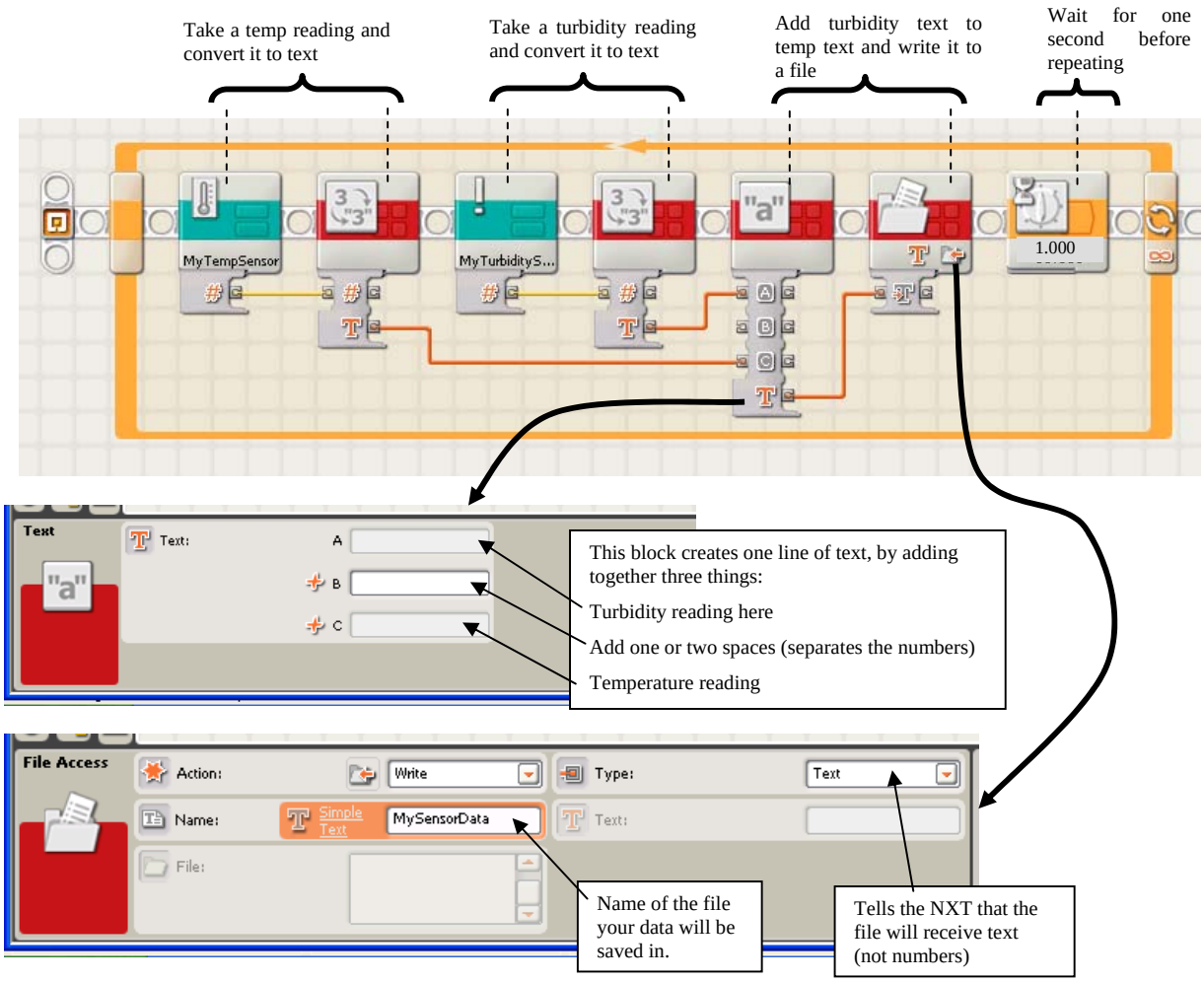
After verifying that your NXT is connected correctly click the Memory tab at the top of the NXT window. Once the memory screen comes up, click the word “Other” on the left side of the NXT window next to the NXT Memory Usage bar. Select the name of your file in the window and click upload. Select where you would like the datalog file saved and click OK. Then close the NXT window.



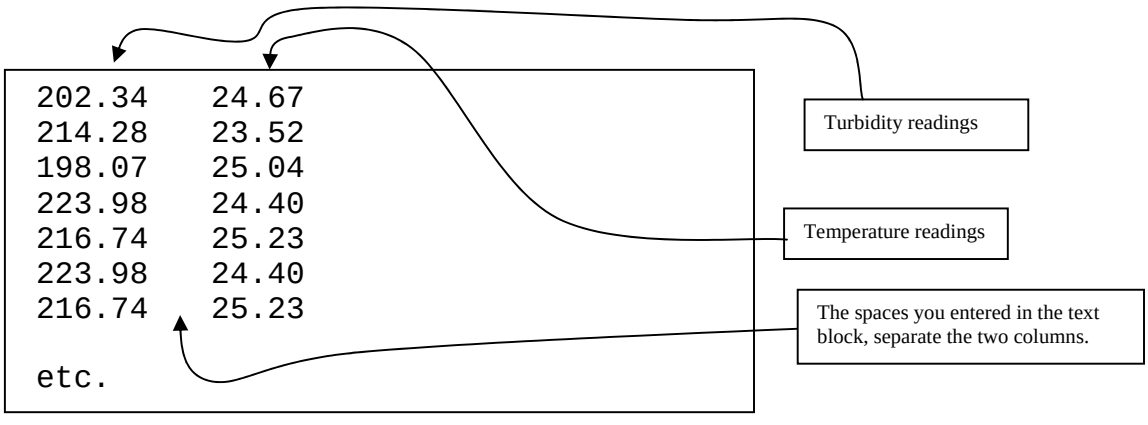
Go to where you saved your data-log file (in this case “MyTempReadings.txt”) and open the file. In the file you should see a column of numbers. These are the temperatures that your NXT sensed at 1 second intervals while your program was running.

If you wish, you can also open your data in Microsoft Excel. To do so, open Microsoft Excel, and select “Open”. Under Files of Type, select “Text files”. Choose the folder on your computer where you saved your file, and it should show up in the list. (In this example, the file would be called “MyTempReadings.txt”.) Click on Open and the Text Import Wizard should appear. Make sure “delimited” is selected, then click on Next. Place a checkmark next to “space”, then click on Finish. This should place your data in columns in an Excel file. Alternatively, simply open the text file, highlight the column of numbers, select “copy”, then go into excel, click at the top of a column, and select “paste”.

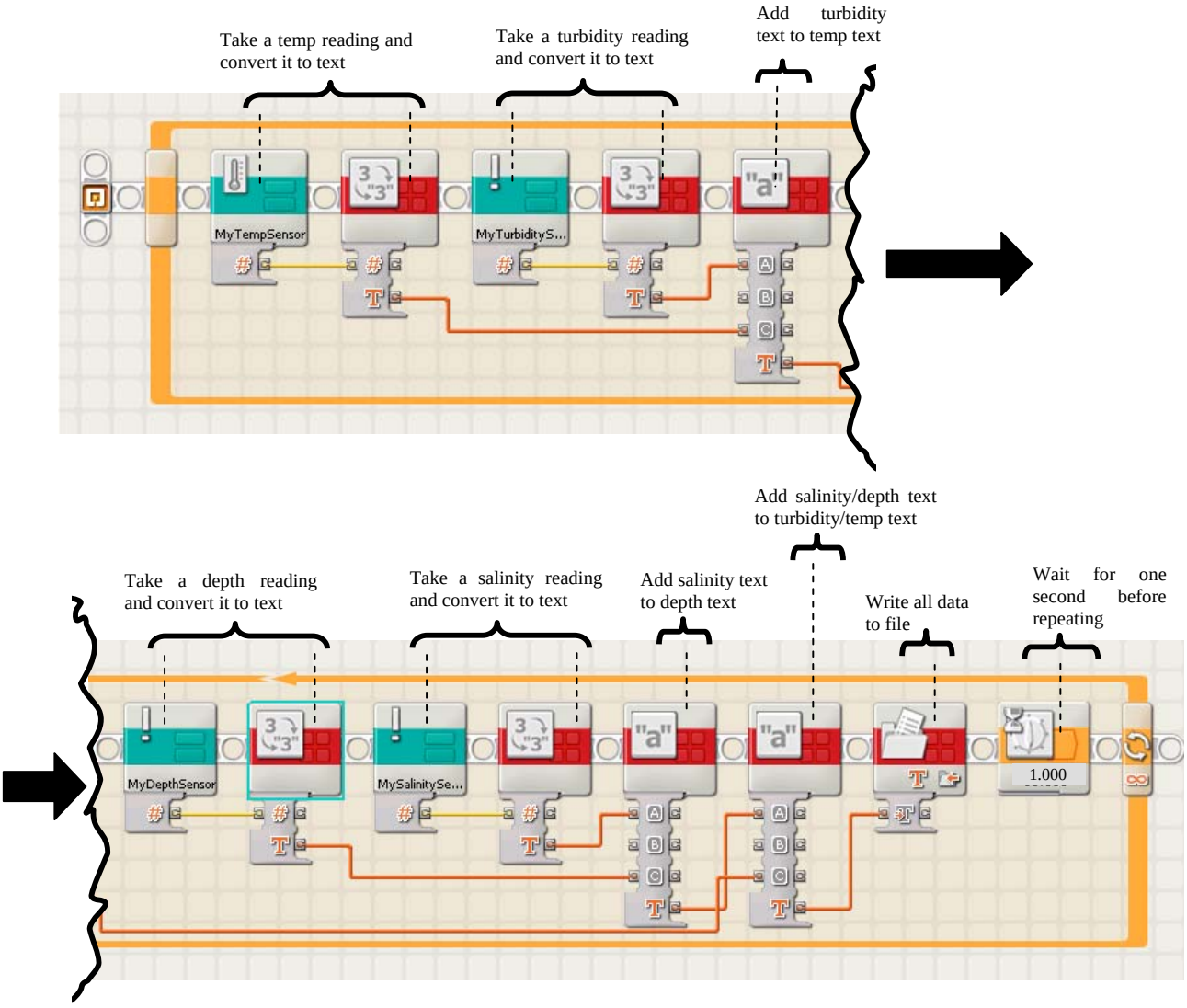
2) Now try writing a more complicated program that will simultaneously record data from two different sensors and save the data in a single file. There are many ways to write this program, but one simple way is as follows:



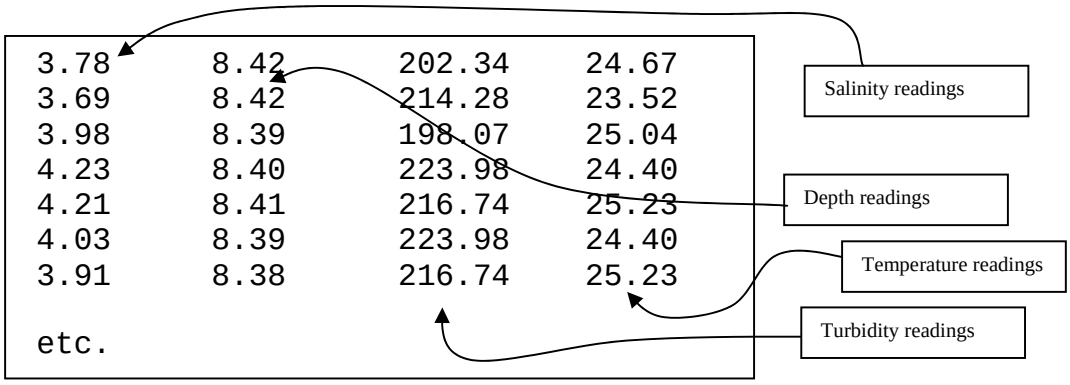
Run the program, then upload the file “MySensorData.txt” from the NXT. It should contain two columns of numbers and look something like this:



3) As a final exercise, try writing a program which simultaneously records data from all four kinds of sensor (temperature, salinity, turbidity and depth), and saves it all into one data-log file. Again, there are many different ways to do this, but one simple way is like this:



Run the program, then upload the file “MySensorData.txt” from the NXT. It should contain four columns of numbers and look something like this:



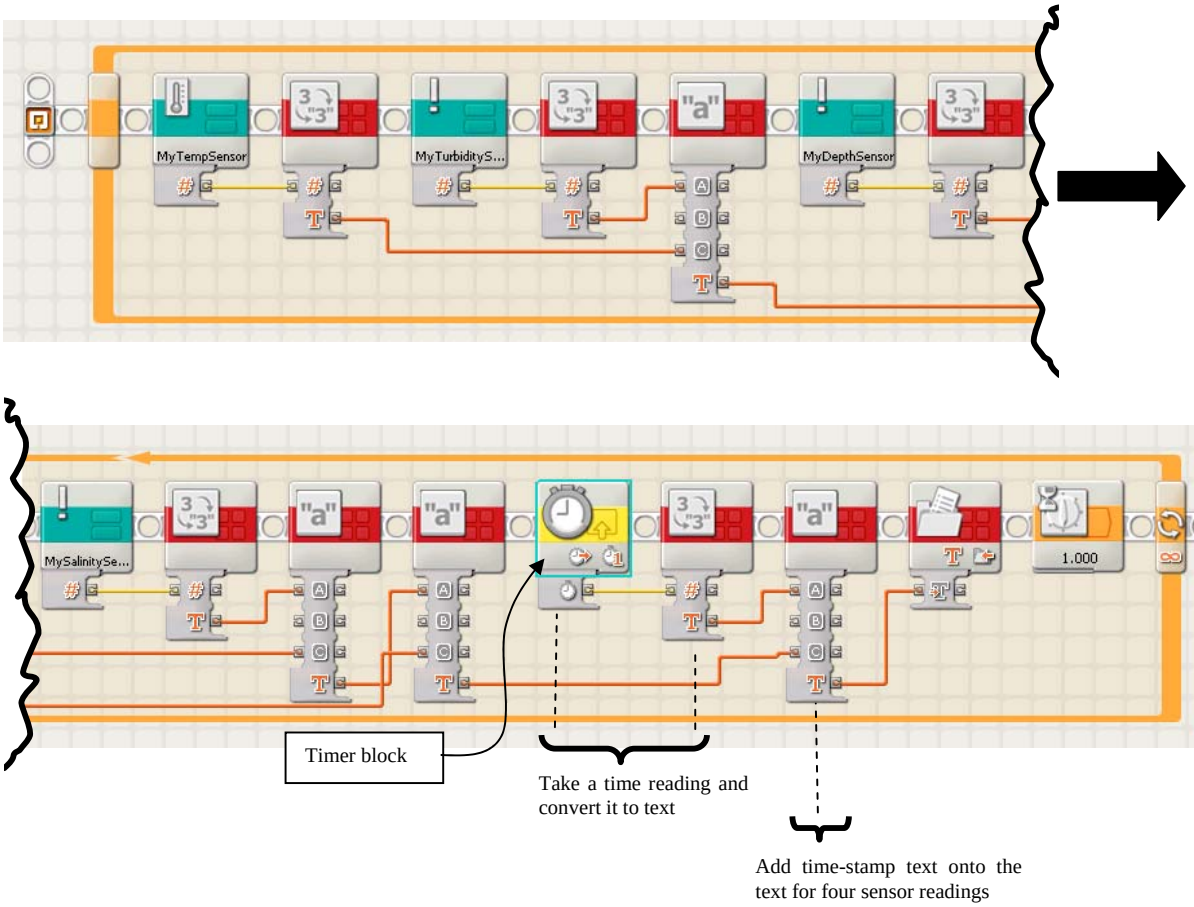
Add-on note: how to add time-stamps to your data-log entries:

The previous program that you wrote, captures one reading from each sensor every second. The program outputs a text-file which has four columns of numbers. Each column represents a different sensor, and each line of numbers represents a particular time – with one line of sensor numbers captured every second.

For some activities, you may wish to add some extra program blocks, so that your program outputs a text-file containing five columns of numbers – where the extra column now represents the time at which each line of sensor readings was captured.

In general this is not usually necessary. Since one line of readings is captured every second, you know that the fifth line of readings happened at five seconds, and the 206th line of readings happened during the 206th second of operation. If you open up your data in an excel spread sheet, then the line numbers in excel will already represent the time-stamp in seconds (or whatever unit of time you choose to iterate your data capture loop with in your program).

However, if you do wish to have your program output a timestamp, the following is an example of how to do this – it will add an additional column of numbers into the output text-file which represent the time in milliseconds since the program began running.



Note: the timer block simply outputs the number of milliseconds which have elapsed since the program began running.

Notes on “My Blocks”

(reproduced from the LEGO Mindstorms Help file)

The My Block Builder lets you take a number of blocks you select in the work area and group them together into your own My Block with a customized icon. For example, a My Block you create called “Robot Motion” might group a Move block (to control your robot’s wheels) together with a Sound block (that plays a sound file when the robot has traveled a certain distance).

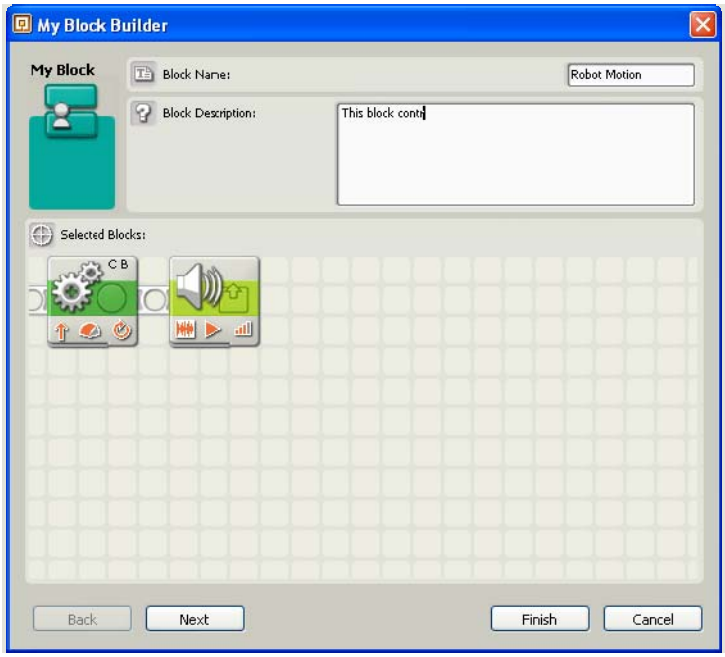
When you create a new program using the same robot, you can just drag the “Robot Motion” block from the Custom palette and you will be done programming this part of the robot’s motion because you already set all of the parameters in the My Block.

Over time you may develop a library of My Blocks that you can use in other programs and trade with other MINDSTORMS users.

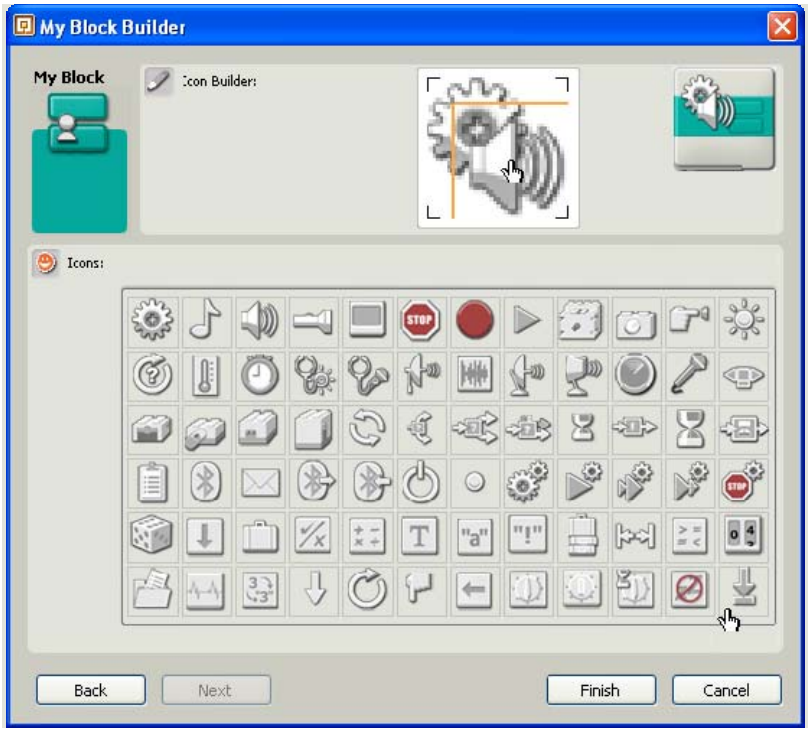
Creating a My Block

1. First select a number of blocks that logically go together. As in the example above, you might choose a number of blocks that control your robot’s movement and choose to group them into a My Block.
2. Select the blocks by placing your mouse pointer to one corner of an imaginary rectangle that will surround all of the blocks you want in the My Block. Hold down your mouse button and sweep the pointer to the opposite corner of the imaginary rectangle, surrounding the blocks. When you release the mouse button, all of the blocks for your new My Block should be selected. Remember that a selection that partly covers a block will include that block in the My Block.

3. With the blocks selected, choose the Make a New My Block command from the Edit menu at the top of the NXT software interface. This will open the first My Block Builder wizard screen (shown below).



4. Give your new My Block a name and write a short description of what it does. For example: “This block controls the motors of a TriBot and plays a tone when the robot has traveled one meter.” Click Next to move to the next step.



5. Next, design your new My Block’s icon. Use your mouse to drag an icon or two into the editing box. Your keyboard's arrow keys will allow you to fine tune the positioning of the icon(s). The example block to the right shows what it will look like when you are done.

Click Drag an icon or two into the editing box at the top of the window. Use your keyboard's arrow keys to adjust each icons final position. The graphic to the right shows what the block will look like. Click Finish when you are done. Your new My Block will appear in your current program and in the Custom palette, which is accessible by clicking the right-most tab at the bottom of the programming palette. (See image below.)

Note: remember that any input or output data wires that crossed the selection frame (that first enclosed the blocks making up your My Block) will be created as input and output plugs on your new My Block's data hub.

If you want to change the available plugs of a My Block, you will have to create the My Block again, this time with the right plugs (and corresponding data wires) extending out of the selected area.

Editing an Existing My Block

If you want to change the performance of a My Block, you can edit the blocks it contains by double-clicking the My Block or by choosing Edit Selected My Block from the Edit menu. If you just want to change a My Block's icon, you should select the block and choose the Edit My Block Icon command in the Edit menu––this will launch the My Block Builder where you can make any changes.

Managing the Custom Palette

You can add and delete My Blocks from the Custom Palette by selecting the Manage Custom Palette command in the Edit menu. To send a My Block to a friend, make a copy of the desired My Block and attach it to an email message. If you receive a My Block from a friend, choose the Manage Custom Palette command and drag the new My Block to the My Blocks folder.

You can create new sub palettes in the Custom palette by creating a new folder in the Block folder. Customize each sub palette's appearance by placing two 45x45 pixel Portable Network Graphics (PNG) files (called PaletteIcon.png and PaletteIconHilight.png) in the sub palette's folder. You can specify a tooltip for each custom sub palette by placing a text file called PaletteIcon.txt (with the tooltip text in it) in the same folder.