Fun with Data

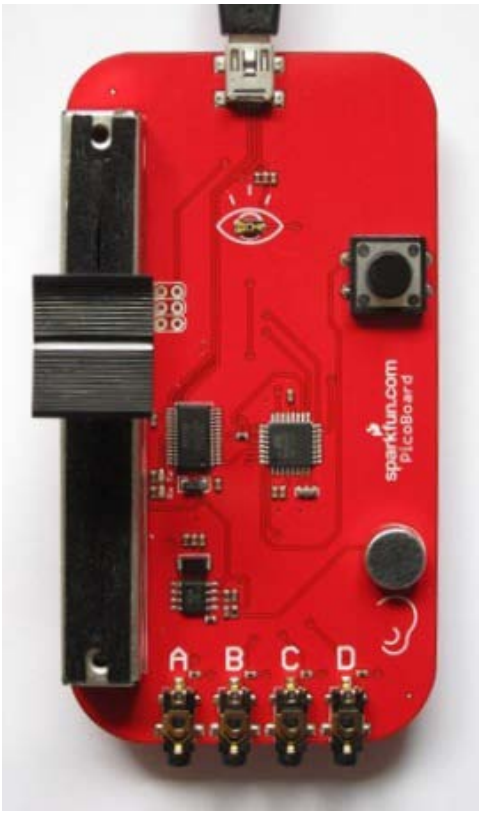
Data collection and investigation with Scratch and a Picoboard

By Myra VanInwegen

Scratch, a programming language and environment from MIT's Lifelong Kindergarten, is a hugely fun way to start learning about computer programming. In addition, it's free, and it runs on pretty much any computer.
Download it from here:

http://scratch.mit.edu/

If you add a Picoboard you can use it to collect data from the real world. There are sensors for light and sound, and the ability to add analogue inputs allows you to attach a wide variety of other sensors. In this report, I describe how to use some Scratch programs that I have written to collect data and investigate it. For example, you could record the sound levels in a classroom for a school day and see which periods are the quietest and loudest. This project requires some familiarity with Scratch, although you do not have to be an expert. You should, however, be familiar with the use of variables in Scratch. The target audience is a teacher in later primary school or early secondary school, although it would also work for a parent to do some maths/science/computing investigations at home.



The Picoboard

Project summary

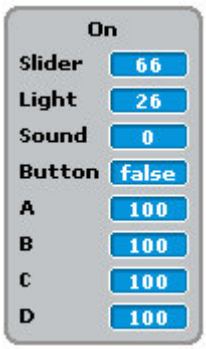
Use the data_collector.sb program to grab a series of data values at regular intervals from one of the sensors on the Picoboard. Use the data_plotter.sb program to plot a graph of the data and to do some basic investigations on it. The point of this project is that graphs are a powerful way of showing data. Once you know how to interpret this sort of graph (that the horizontal axis represents time and the vertical axis represents the size of the data value) many aspects of the data almost jump out at you. These two programs allow you (and your students) to explore data that you have collected yourself. Hopefully this will give the results more meaning. The programs are written in Scratch as this is a language that many of the students will have seen. Although I don't expect the students to understand all the aspects of what I've written they should be able to understand much of it, and it can serve as an example of different aspects of Scratch programming (such as using list and pen code blocks). In fact the aspect of the code that is most difficult to understand is probably the scaling involved in the plotting, which could itself be the subject of an interesting lesson.

This has turned out to be a fairly long document! I suggest that you do not read it through all at once. Just download the data_collector.sb and data_plotter.sb programs and start playing. If you

find that the programs aren't doing what you expect, or you would like to explore more of their capabilities, then by all means read on!

Data collection using the Picoboard

The Picoboard is a circuit board about the size of a credit card. On it are a collection of sensors and other inputs: a slider, a light sensor, a sound sensor, a button, and four analogue inputs. The Picoboard attaches to your computer by a USB cable. Scratch is designed to work with the Picoboard, so it should be easy to set up (although on some systems you do need to do a bit of configuration). The best way to check that it is working is to bring up the ScratchBoard Watcher. To do this, select the Sensing code blocks (by click on the Sensing button in the top left corner of your Scratch window), and then right-click on the "<slider> sensor value" code block that is near the bottom of the left panel. Choose the "show ScratchBoard watcher" menu item. (In my text descriptions of Scratch code blocks I am using <> to indicate a bit of the code block that can change, with a sample value inside the <>.) The ScratchBoard watcher is a box that shows the current values of all the sensors on the Picoboard. If it is working, you will see a variety of values in the display, and some of them should be changing (the sound input will change quickly in most rooms). If it is not working, all the values will be 0.



If you haven't played much with the Picoboard before, it's a good idea to familiarise yourself with it. To use Picoboard inputs in your program, drag the "<slider> sensor value" or "sensor <button pressed>?" block into your scripts. The "<slider> sensor value" block has rounded ends to show that it can be used anywhere you would put a number, for example in the x or y positions in a "go to x: <0> y: <0>" block. The "sensor <button pressed>?" block is hexagon shaped, to indicate that it is a Boolean value (either true or false) and can be used in any hexagon shaped hole, such as in the test of an "if" block. To make a simple Picoboard-based program, you can use the "<slider> sensor value" block (with the slider option selected) to set the x position of the Scratch cat, as shown to the left. Note that the cat only moves around a little bit. This is because all the Picoboard readings are between 0 and 100, so your cat will only move between the centre of the stage (x = 0) and a point a bit further to the right (x = 100). Can you scale and translate the slider input so that the cat moves all the way to the left and right edges of the stage depending on the slider position?

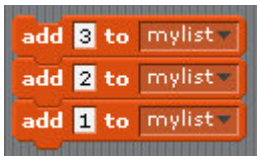
To collect data using the Picoboard, you can use the Scratch program data_collector.sb, which you can download from same place that you got this document. This is a relatively short program that makes collecting data from the Picoboard really easy. Once you have loaded up the program you need to choose the sensor from which to gather data. Make sure that the cat sprite is selected and click on the Scripts tab to see the code. Near the bottom there is a "<slider> sensor value" block (this is the blue block on the

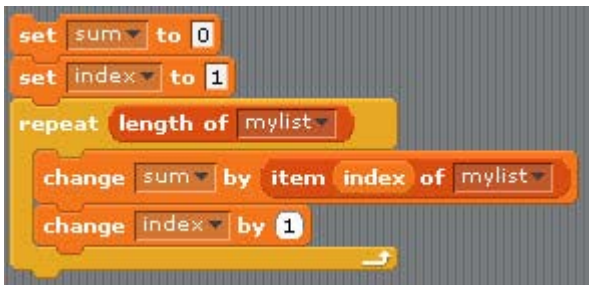
picture above, note that the light sensor is selected in this screenshot). Use the menu to select what kind of data you want to record.

When you click the green flag the cat will ask you how many data readings you want to take and how long you want your test (experiment) to run. For flexibility, you enter the time in hours and minutes. So if you want your test to last for 1½ hours, you could either type in 1 hour and 30 minutes, or 90 minutes if you prefer. You can make the test as long as short as you like. You can even make it less than one minute long by entering a decimal value, e.g. 0.5 minutes if you want a 30 second test. The cat will then ask you what time it is. This information will be stored along with the data, so that later on when we look at the data we will be able to determine at what time events happened. Again you enter the time in hours and minutes. Please use 24 hour time, so for example, 3pm should be entered as 15 hours, 0 minutes.

After you have entered all the information the cat asks for, the program divides the time up into equal intervals, and after every interval it takes a reading from the sensor you have selected. It saves all these readings into a Scratch list called “data”, which is displayed in a box on the Scratch stage.

If you have not investigated Scratch lists before, it’s worthwhile having a look at them now. You create a list by selecting the Variables code blocks (by clicking the “Variables” button in the upper left corner of the Scratch window), then clicking on “Make a list”. As with a normal variable, you give it a name, and you can choose to have this list accessible only by this sprite or by all sprites. Let’s say you’ve called your list “mylist”. When you create a list a box pops up on the stage to show you what’s in it. Initially “mylist” is empty. You can add new items to the end of the list using the “add <thing> to <mylist>” code block. Bring it over into the scripts area and replace “thing” by whatever it is you want to put into your list, and make sure that “mylist” is selected as the list. Or if you like you can click on the “+” in the lower left corner of the list box on the stage and then type something into the empty box that appears in your list. Once you have added a few items to it you can see that “mylist” is basically a numbered list. Each number on the left is called an index (with plural *indices*), and the item to the right of that index is the item stored at that index (also called the value of the list at that index). You can give new values to existing list items using the “replace item <1> of <mylist> with <thing>” code block, and you can use other code blocks to insert new items into the list (at a place other than the end of the list) or to delete items from the list. Have a play to see how these work to add or remove things from lists.





You can select an element of a list with the “item <1> of <mylist>” block. This has rounded ends and thus can be used in any code block where you find “thing” or a number. A common way to access lists is to use a loop to run through all the elements of a list. For example, the bit of code shown will add up all the items in “mylist”.

Now back to collecting data using data_collector.sb... Note that after the test/experiment is run, the length of the "data" list is always two greater than the number of readings you asked for. This is because my program stores two extra pieces of information in the list. The item at index 1 is the time of day when you started your data collecting (the number stored here is the number of minutes past midnight). The item at index 2 is the number of seconds between successive readings. This information is used later (in the program data_plotter.sb) to allow you to find out when a particular thing that interests you happened. Doing this can provide quite a bit of insight into the data. For example, say the data is a record of the noise levels in a classroom over the course of a school day. If you see a big dip in the noise levels, you might investigate this and discover that this was during lunch time, when no one was in the classroom!

When the data is all collected, the cat says "I am done! Now you can save the data by right-clicking on the "data" box." Clicking your right mouse button on the box containing the data list brings up a menu. If you choose "export" from this menu you can save the recorded data (along with its time info) into a file. This is a text file that you can look at with a simple text editor such as Notepad on Windows computers.

After you save the data list to a file, you are done with data_collector.sb. Now let's see what you can do with the data!

Data display

Now, load up the program data_plotter.sb, which is also available from the same place you got this document. This is a much larger program than the data collector. It includes the features that I found most useful in playing around with the data I collected. The first thing you need to do when you start up the program is to load in a data set that you have collected using the data collector program. You do this by right-clicking on the "data" box on the Scratch stage, selecting "import" from the menu, and choosing the file in which you saved your data.

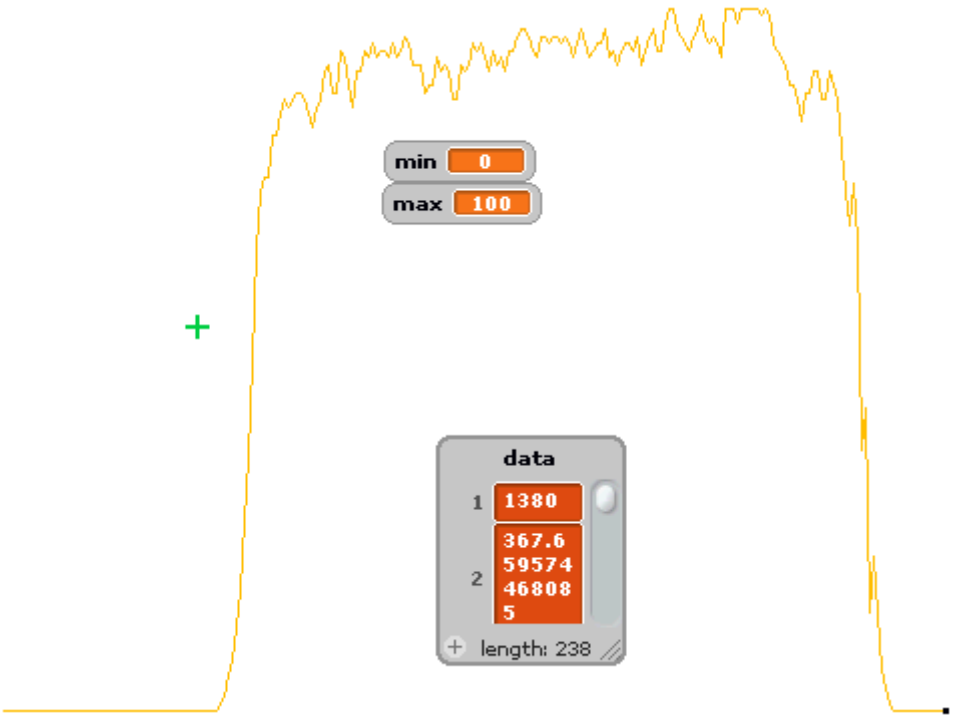
There are three ways to configure the plotting of data. To get to the first two, select the cat sprite and click on the Scripts tab. Just underneath the "when green flag clicked" block is a block saying "set <flip_y> to <false>". If "flip_y" is set to true, the graph drawn will be flipped upside down, so that a larger input value results in a value lower on the screen (a smaller y value) and vice versa. This option is mainly useful for when your input comes from a thermistor (since many thermistors increase their resistance when the temperature goes down). For most purposes you'll want "flip_y" set to false, so a larger input value results in a point higher on the screen.

The second configuration item determines what sort of plotting you want. Look for the last block in the cat's main script. This is a broadcast block, which can broadcast one of two messages: "plot_it_simple" or "plot_it_better". If you choose "plot_it_simple", then when this script is run the message is picked up by the plot_simple sprite, which draws the graph on the stage in a very simple way. It ignores the timing information that we've put at the beginning of the list and then goes through the list, plotting each point without scaling it. So if the item at index *i* is *v*, we'll put a point at (x=*i* ,*y*= *v*) on the stage(for an *i* of 3 or greater). This is not very good, as it all the data is bunched up into the first quadrant of the stage

(where x and y are both positive). If you choose “plot_it_better”, the message will be picked up by the plot_better sprite, which draws the graph on the stage in a much nicer way. It finds the minimum and maximum values of the data and then plots the graph, scaling both x and y so the stage is filled by the data. This spreads out the data so you can see it much better. The main reason I have included the plot_simple sprite is to give an example of a very basic plotting script, in case someone wanted to have a go with their students at creating a plotting script that scales the data in a similar way to plot_better.

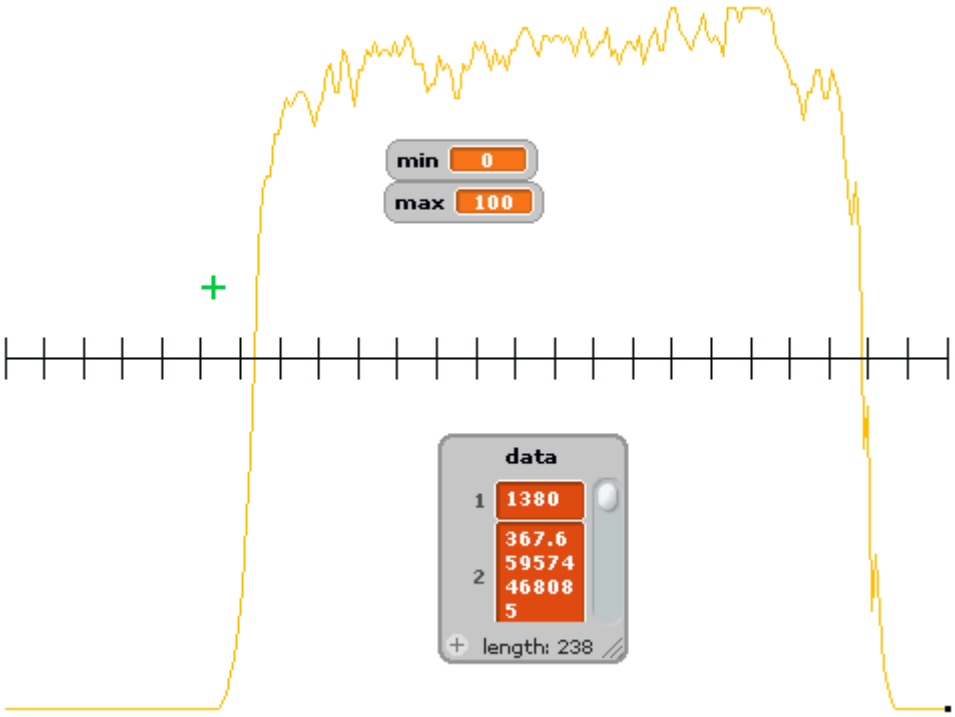
The final configuration item is relevant only if you are using the plot_better sprite. Click on this sprite to select it, then click on the Scripts tab. The main script here is triggered by a “when I receive <plot_it_better>” block. Near the top of this script is a block saying “set <manual_set_bounds> to <false>”. If the variable “manual_set_bounds” is set to false, the program will find the largest and smallest values in the data and use this information to scale the graph so it will occupy all the vertical space on the display area. If it is set to true, the blocks “set <min> to <0>” and “set <max> to <100>” will be executed. This allows you to set what values would end up at the top and bottom of your display area. This is mainly useful if you want to plot more than one graph on the display area. For example, if you’ve recorded the light levels at a window on different days, you’ll want to make sure that they are both displayed with the same vertical scale factor, so it’s immediately apparent if one day was duller than the other.

After this bit of configuration, click the green flag to plot your data. A sample plot is shown below. This is a plot of data from the light sensor, which was pointed out a west-facing window, and 236 data values were collected over 24 hours.



If you have chosen the “plot_it_better” option, you will get a green cross appearing on the stage. This is the “inverter” sprite. It allows you to get information about the graph that has been drawn. If you use your mouse to move the green cross to a bit of the graph that interests you, then press the “I” key on the keyboard (I for inverter), you will see five variables show up on the stage. They will be set to indicate the data value and time corresponding to that bit of the graph. The time is displayed in days, hours, minutes, and seconds and thus uses four variables. The time is computed from the x value at the centre of the green cross and the “seconds_per_reading” data item that was stored in the list along with the data proper. The time shown by the inverter can be either the time since the beginning of the test or the actual time of day when that reading was taken. This is set by changing the value assigned to “show_time_of_day”. Partway down the script of the inverter you’ll find the code block “set <show_time_of_day> to <false>”. If the value assigned to “show_time_of_day” is true, then the start time of the test is added to the time since the beginning of the test, so the time variables show you the time of day for this bit of the graph. (In this mode, if the time variable for the days is greater than 0, this means that the data was collected on a different day from the day in which the test was started.) If the value assigned to “show_time_of_day” is false, then the time values just show the elapsed time since the start of the test.

The horizontal_line sprite draws a horizontal line with ticks indicating time intervals. It is set into motion by pressing the “H” key on the keyboard. You are asked for the value at which to draw the line. This refers to the values in the data list so, for example, if 24 is between the min and max of the data values in your list and you give 24 as the data value at which to draw the line, the line will be drawn at the same y value that a data value of 24 would be drawn at, not at y=24.You are also asked for the time interval (in hours, minutes, and seconds) between ticks on the line. There is no sprite to draw a vertical line, mainly because the Picoboard readings are difficult to correlate with any concrete measurements in the real world. For example, decibels can be used to measure loudness, but the Picoboard measure of loudness is not directly related to decibels. What is important about Picoboard readings is not the actual value of them, but the pattern: which readings are higher than others. For this reason, I haven’t provided a vertical scale. Below I show the daylight graph with a horizontal line at value 50 and with ticks for each hour.



Because the stage isn't cleared before a graph is plotted, you can use this program to draw graphs of more than one data set on the display area. Each time you click the green flag to plot your data, a different colour is chosen for the graph. On the other hand, if you want only one graph plotted at a time, you can use the "clear" code block (one of the Pen code blocks). If you are going to show more than one graph at a time, it is most useful if the readings were taken over the same length of time (say over a 3 hours period) so that the horizontal scale is the same. They do not have to have the same number of readings or interval between readings. The data sets do not need to have the same maximum and minimum values, although if they don't they will by default be scaled differently along the y axis. To prevent this, use the variable "manual_set_bounds", as described above, to set the max and min values so that the min is less than or equal to any value in all the data sets and max is greater than or equal to all the values. The easy way to find the overall max and min values is to first plot the graphs without manually setting the bounds. Then after each graph is plotted, the max and min values for the graph will be in the variables "max" and "min". If these variables aren't displayed on the stage, you can make them show up by selecting the Variables code blocks, then putting a tick next to the "max" and "min" variables. Or, since this is Picoboard data, you could just set min to 0 and max to 100.

So, what do you do with all this?

Probably once you get playing with this you will come up with your own ideas. Here I've collected together a few that might entertain and enlighten.

- Record the sound levels in a classroom to find the loudest/quietest times of the day.
- Record the light levels with your Picoboard stuck to a window looking outside. Do this on several different days (even better yet at different times of year) and compare. Ca you see that the daylight periods are much shorter in the winter? Can you see the effects of daylight savings time in the data?

- Attach a thermistor to the one of the analogue inputs of the Picoboard and measure temperature. But this is a subject of another project *Watching Ice Melt (Electronically!)*

Most of all, have fun with your data!