|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Scratch Data Logger**  Using the pen feature in Scratch and the PicoBoard, we will show you how to turn Scratch into a Data Logger / Graphing application.  Using this guide, we hope you’ll come up with other ways to integrate graphing, data collection, and scientific inquiry into the Scratch environment. | **graph.gif** | | |  |  | | --- | --- | |  |  | | **ELECTRICAL PROTOTYPING** |  | | **ROBOTICS** |  | | **SOLDERING** |  | | **PROGRAMMING** |  | | **DIY** |  | |

|  |
| --- |
| MATERIALS LIST |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | * Computer | * Scratch | * PicoBoard |  |  | |

|  |  |
| --- | --- |
| **STEP 1: Creating your sprites**  For this activity, we are going to delete the default Scratch cat. Right click on the cat in the lower-right and select delete.  Now, on the bar just above this, click on the paintbrush or the “Paint new sprite” icon.    Use this paint window to create a new sprite that will be our *drawing cursor*. You can make your sprite any shape you like, but I suggest keeping it simple - something like a dot.  **STEP 2: Setting the starting position and incrementing the drawing cursor.**  The blocks here are pretty simple. We want the cursor to start on the left-most edge of the screen. This corresponds with x: -240 and y: 0.  We also want the cursor to start tracing across the screen - one step at a time. We use the “**change x by [ ]**” block. This increments one step at a time.  graph1.gif |  |



|  |
| --- |
|  |

|  |  |
| --- | --- |
| **STEP 3: Moving the sprite based on the sensor value.**  Next, we will use the graph1.gifblock. Insert the sensor value block. You’ll find the sensor value block under the **Sensing palate.**  **STEP 4: Run**  Run the script by either double-clicking on the script, or clicking on the green flag. Move the slider up and down, and you should see the pen trace follow on the screen.  **STEP 5: Making the code better?**  Notice what happens when the sprite hits the edge of the screen. Let’s make the sprite return to the left side of the screen. We’ll use an **if block** for this.  Let’s catch the case when the **x position** is equal to 240. When this happens, we want to reset the x position to -240 and clear the screen. The **x position** block can be found at the bottom of the ***Motion palate***.  String these blocks together to look like this:  **graph2.gif** | graph1.gif  **graph2.gif**  **graph2.gif**  **graph2.gif** |

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| TAKING IT FURTHER |
| * The sensor values range from 0 to 100. Notice how the sprite only moves from the middle of the screen to about ¾ up the screen. To make this better, let’s scale the sensor value to the full range.  |  |  | | --- | --- | | * The y-value needs to scale from a range of (0 - 100) to a range of (-180 - 180). This equates to a scaling factor of 3.6. Multiply the sensor value block by 3.6. This is only part of the scaling. This will give us a range of 0 to 360, but we need it to be a range of -180 to 180. To adjust our range, we’ll add an offset of -180. * Change the background to a graph paper with equal marks and ticks. Or, use the sprite to draw a grid in the background before the **forever loop**. | graph5.gif  graph6.gif |  * Add a second sprite that draws a trace in a different color based on one of the other sensors. |

