**Code conversation level 1 - Solar energy meter**

**Code for the program is here:** https://tinyurl.com/skxfbrjw

**Describe what the whole program is doing in a couple of sentences:**

**Fill in the blanks**

| ***Line*** | ***Code*** | ***What is this line doing?*** |
| --- | --- | --- |
| 1 | from microbit import\* |  |
| 2 |  |  |
| 3 | P0\_MAX = 812† | This is a constant which sets the maximum value expected from pin PO, and this is the value that represents the solar store being 100% charged. You might have to experiment with the value here to get the best performance from your barchart. |
| 4 | This function will determine the bargraph display on the micro:bit |  |
| 5 | def barchart††(y, v, vmax): | Defines a Python function called barchart () with the parameters y,v,vmax, which draws a horizontal line on the display by turning on a different number of LEDs  ‘y’ is the position on the display where the barchart will appear  'v' is the value to display  'vmax' is the maximum value to expect |
| 6 | v = min(v, vmax) | The value is capped (with min()) i.e. the value of v cannot exceed vmax, it will be whichever is lower, the reading i.e. v or vmax which is the maximum permitted. |
| 7 | leds = int(v \* 5 / vmax) | In order to display the correct number of lit LEDs this variable is used which is scaled (with the divide) to turn on between 0 and 5 LEDs. |
| 8 | for x in range(leds): |  |
| 9 | display.set\_pixel(x, y, 9) | display.set\_pixel(x,y,value) - this inbuilt function sets the brightness of the LED at row x and column y to value, which has to be an integer between 0 and 9. In this case the brightness is always set to the maximum 9. |
| 10 |  |  |
| 11 | # main program |  |
| 12 | while True: |  |
| 13 | reading = pin0.read\_analog() | Sets variable for the reading. The function pin0.read\_analog gives a number between 0 and 1023 depending on the voltage at pin0 (which is the Charge% pin on the Solar Store. |
| 14 | display.clear() |  |
| 15 | barchart(4, reading, P0\_MAX) | Call barchart function with arguments (4,reading, PO\_MAX) - reading here refers to the variable set at line 13 which is the “reading” from the Charge% pin on the Solar Store. |
| 16 |  |  |
| 17 | if button\_a.was\_pressed(): |  |
| 18 | pin2.write\_digital(0) # off | Removes the voltage on pin P2, which is connected to the Enable pin of the Solar Store, and this will remove the power to the right hand 3V pin on the Solar Store which stops the fan. |
| 19 | if button\_b.was\_pressed(): |  |
| 20 | pin2.write\_digital(1) # on | Restores the 3V voltage on pin P2 which enables the output and the fan starts. |
| 21 |  |  |
| 22 | sleep(1000) # 1 second |  |
|  |  |  |

†The value of PO\_MAX is 812 because the voltage stored in the Supercapacitor is halved before measurement, and therefore has a maximum of 2.5V. On a micro:bit, this 2.5V reading results in an analog value of around 812.

††barchart() draws a horizontal line on the display, by turning on a different number of LEDs. 'y' is the position on the display where the barchart will appear, 'v' is the value to display, and 'vmax' is the maximum value to expect. The value is capped (with min()) and scaled (with the divide) to turn on between 0 and 5 LEDs. The main program loops round forever reading the Charge% pin, displaying a barchart, and checking the buttons. Button A = fan off, Button B = fan on.

**Extra detail about the program and circuit:** Pin P0 of the micro:bit is connected to the Charge% pin on the Solar Store. The P0\_MAX constant sets the maximum value expected from pin P0, and this is the value that represents 100% charged. You might have to experiment with the value here to get the best performance from your barchart.