**Code conversation level 2 - Intelligent cooling fan**

**Code for the program is here:** https://tinyurl.com/2wtkdh2d

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

The intelligent cooling fan turns on if the ambient temperature is warm and the solar store is more than half full, then off if the temperature is cold or the solar store is not full enough. In addition the program puts a message on the micro:bit which says YES when the fan is on and NO when the fan is off. Buttons A and B are provided for an override that allows you to request the fan manually. B overrides and A cancels the override.

**Fill in the blanks**

| ***Line*** | ***Code*** | ***What is this line doing?*** |
| --- | --- | --- |
| 1 | from microbit import |  |
| 2 |  |  |
| 3 | CHARGED = 818/2 | sets a variable called CHARGED. CHARGED is the maximum charge on the supercapacitor (approx 818) divided by 2. This means that as long as the solar store is holding half charge you can use some of it, rather than waiting for a full charge. |
| 4 | DISCHARGED = 220 | sets a variable called DISCHARGED. DISCHARGED is a low reading, which equates to the solar store being about a quarter charged. |
| 5 | HOT = 23 |  |
| 6 | COLD = 20 |  |
| 7 | override = False |  |
| 8 | temp = 0 |  |
| 9 |  |  |
| 10 | def read\_temp(): | defines a Python function called read\_temp () |
| 11 | global temp | specifies that the temp variable that is set inside this function should be available to the rest of the programme i.e. globally and not just inside the function. |
| 12 | temp = temperature() |  |
| 13 | if override: |  |
| 14 | temp = HOT |  |
| 15 |  |  |
| 16 | def fan\_needed(): |  |
| 17 | return stored >= CHARGED and temp >= HOT | Returns the value of stored>=CHARGED (which is max charge/2) and temp>=HOT |
| 18 |  |  |
| 19 | def fan\_not\_needed(): | defines a Python function called fan\_not\_needed(). The function decides when the cooling fan is not needed only when the charge in the Solar Store is low or temperature is cold. |
| 20 | return stored <= DISCHARGED or temp <= COLD |  |
| 21 |  |  |
| 22 | def fan\_on(): |  |
| 23 | pin2.write\_digital(1) | pin2 is switched to 1 i.e. the “Enable” pin on the Solar Store is activated so that any stored energy can be used by the attached device. |
| 24 | display.show(Image.YES) |  |
| 25 |  |  |
| 26 | def fan\_off(): |  |
| 27 | pin2.write\_digital(0) | pin2 is switched to 0 i.e. the “Enable” pin on the Solar Store is de-activated so that any stored energy cannot be used by the attached device. |
| 28 | display.show(Image.NO) | inbuilt function which shows a specific image on the micro:bit, in this case the word NO to mean that the fan is off. |
| 29 |  |  |
| 30 | # main program |  |
| 31 | while True: |  |
| 32 | # sensing |  |
| 33 | stored = pin0.read\_analog() | sets the value in the variable stored to the reading from pin0, which is the pin attached to the solar store. |
| 34 | if button\_a.was\_pressed(): |  |
| 35 | override = True |  |
| 36 | if button\_b.was\_pressed(): |  |
| 37 | override = False |  |
| 38 | read\_temp() | call read\_temp function |
| 39 |  |  |
| 40 | # control |  |
| 41 | if fan\_needed(): |  |
| 42 | fan\_on() | call function fan\_on() (as defined earlier in programme) |
| 43 | elif fan\_not\_needed(): |  |
| 44 | fan\_off() | call function fan\_off() |
| 45 |  |  |
| 46 | print(stored, temp) | the current reading for stored and temp is stored in the program |
| 47 | sleep(1000) |  |
|  |  |  |