1. Etch the SD card using "Raspberry Pi Imager" from https://www.raspberrypi.org/

[note: you have to do these next steps from the actual Pi, with keyboard/mouse/monitor, because SSH is disabled by default]

2. Boot up the Pi, set up your language, wifi password, etc, and let it download all the updates, then reboot

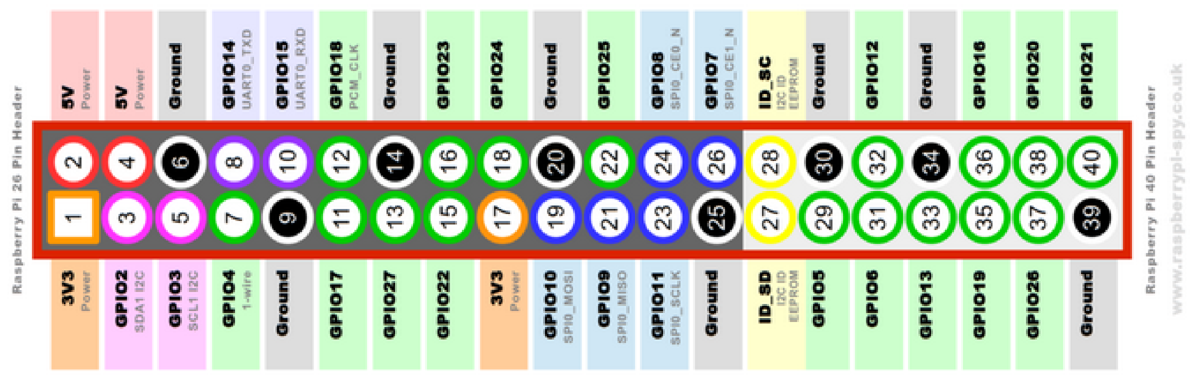
3. Go to the App Menu --> Preferences --> Raspberry Pi Configuration --> Interfaces tab.
   a. Enable the "1 wire" interface
   b. Enable SSH

4. Make the following wire connections:
   a. BLUE to pin 39 (GND)
   b. RED to pin 1 (3.3v)
   c. YELLOW to pin 7 (GPIO4/1-wire)
   d. 4.7kohm pullup connected between YELLOW and RED

Here's the GPIO header pinout. You can identify which pin is pin #1 on the PCB by finding the SQUARE pin on the bottom side of the board.

Raspberry Pi 40-pin GPIO Header

The majority of Raspberry Pi models use the same 40-pin header :



5. Reboot the Pi

[now you can use KVM or SSH]

6. Download the project files:

```
pi@raspberrypi:~ $ git clone https://github.com/abelastley/PiTempSensor.git
```

7. To verify that the physical connections are proper:

```
pi@raspberrypi:~/PiTempSensor $ ./show_connected_devices.sh
28-00000b9034b3
28-00000b90c6ac
#I have 2 temperature sensors hooked up
```

Note: if you don’t see any devices, verify that the 1-wire interface is enabled and the driver is loaded:

```
pi@raspberrypi:~/PiTempSensor $ lsmod | grep w1
w1_therm               16384  0
w1_gpio                16384  0
wire                   45056  2 w1_gpio,w1_therm
#The “w1_gpio” line means 1-wire is enabled.
#The “w1_therm” line means the kernel has loaded the driver for DS18B20
```

8. Run the main python script. If desired, you can place the temperature sensors in ice water and/or boiling water to verify functionality and accuracy:

```
pi@raspberrypi:~/PiTempSensor $ python tempsensor.py
found 2 sensors
Output file does not exist. Creating new one...

Sensor 0
temperature 206.375
Sensor 1
temperature 32.225
```

9. Verify the data was written to the CSV file:

```
pi@raspberrypi:~/PiTempSensor $ tail TemperatureData.csv
Date and Time,Sensor 0 Reading (degrees F),Sensor 1 Reading (degrees F),
05/28/2020 23:29:46,206.375,32.225,
```

10. If desired, this script will configure cron to run the python script automatically once every minute. It needs to be run as sudo.

    This will stick across power cycles, so once this is run, you can unplug the Pi and plug it back in, and the python script will be run automatically every minute with no user input needed.

```
pi@raspberrypi:~/PiTempSensor $ sudo ./schedule_cron_job.sh
```

You can verify that it worked with

```
pi@raspberrypi:~/PiTempSensor $ tail -f TemperatureData.csv
05/29/2020 13:13:01,70.475,70.3616,
05/29/2020 13:14:01,70.475,70.3616,
05/29/2020 13:15:01,70.475,70.3616,
05/29/2020 13:16:01,70.475,70.3616,
05/29/2020 13:17:01,70.475,70.3616,
05/29/2020 13:18:01,70.475,70.3616,
05/29/2020 13:19:01,70.475,70.3616,
05/29/2020 13:20:01,70.475,70.475,
05/29/2020 13:21:02,70.5866,70.475,
05/29/2020 13:22:01,70.475,70.475,
#you should see a new line appear every minute
```

11. Questions, email Abel Astley at abel.astley@gmail.com