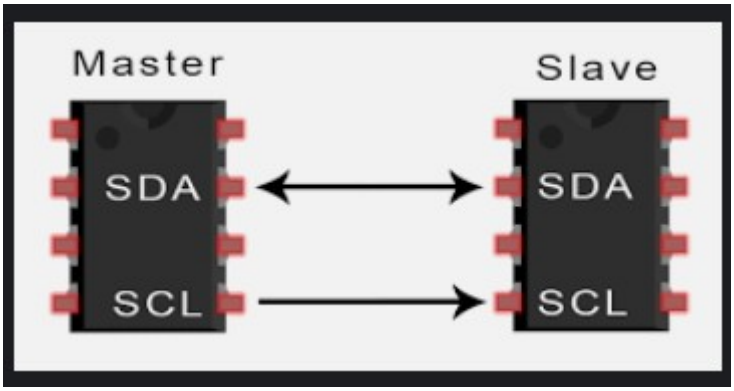
I2C and the DHT2x temperature + humidity sensor

First, learn a little about the very popular widely used I2C (Inter-Inter Connect) 2-wire protocol that drives chip like this one!

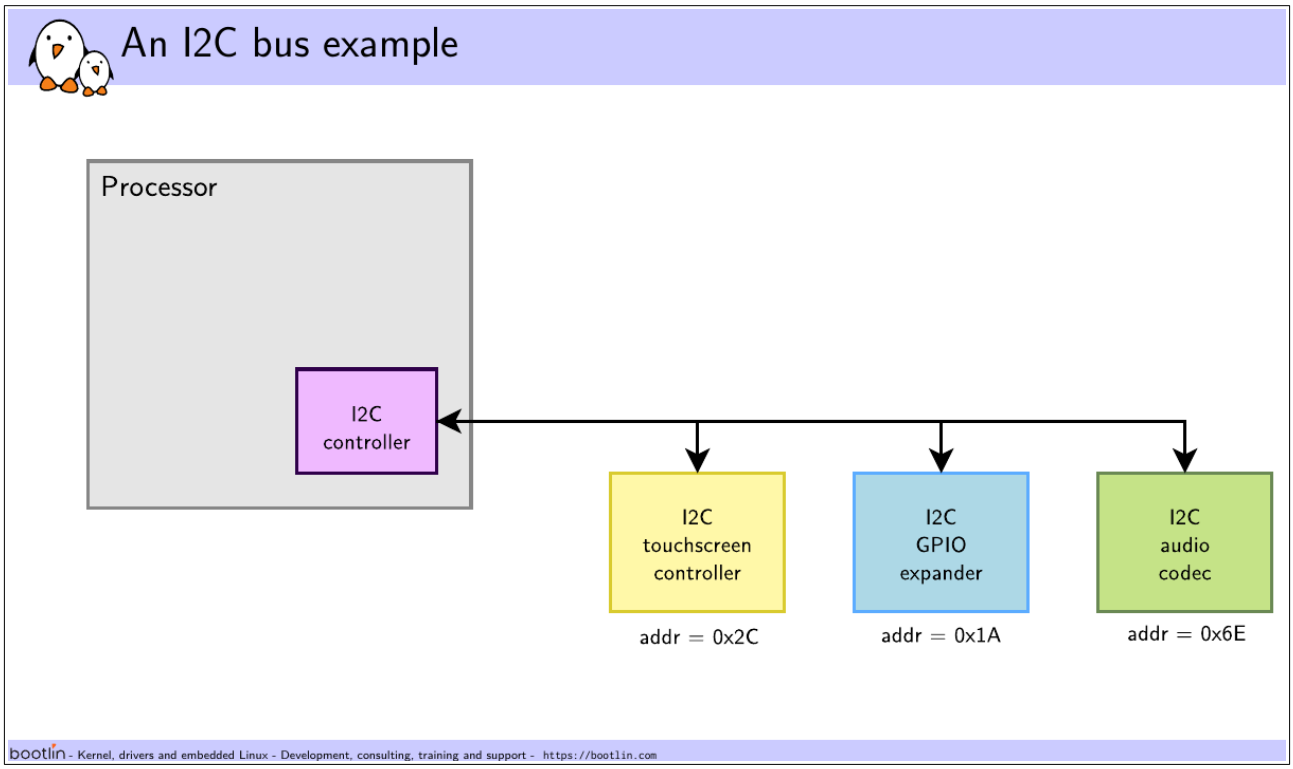
Refer:
Linux Kernel and Driver Development, Bootlin (aka Free electrons): linux-kernel-and-driver-dev.pdf : pg 172 – 185

Ref (for notes following):
https://learn.sparkfun.com/tutorials/i2c/all



<Master/Controller> **<Client chip or peripheral>**

- simple and efficient; only 2 wires (SCL – Serial CLock, SDA – Serial DaTa)
- master / controller initiates communication with the slave or client chip
- I2C controller is usually part of the SoC or processor
- 'Each slave device is identified by a unique I2C address. Each transaction initiated by the master contains this address, which allows the relevant slave to recognize that it should reply to this particular transaction.'



I2C Address List

- Supports
  - multiple controllers (unlike SPI)
  - multiple clients - up to 1008 peripherals (client chips)!
  - clock speeds
    - 0 to 5 Mhz (original I2C)
    - 10 KHz to 100 KHz (Intel's *System Management Bus (SMBus)* version; more controlled protocol)
- 7 bit addresses for addressing clients; implies client addresses range from 0 to 127 (0x0 to 0x7F).

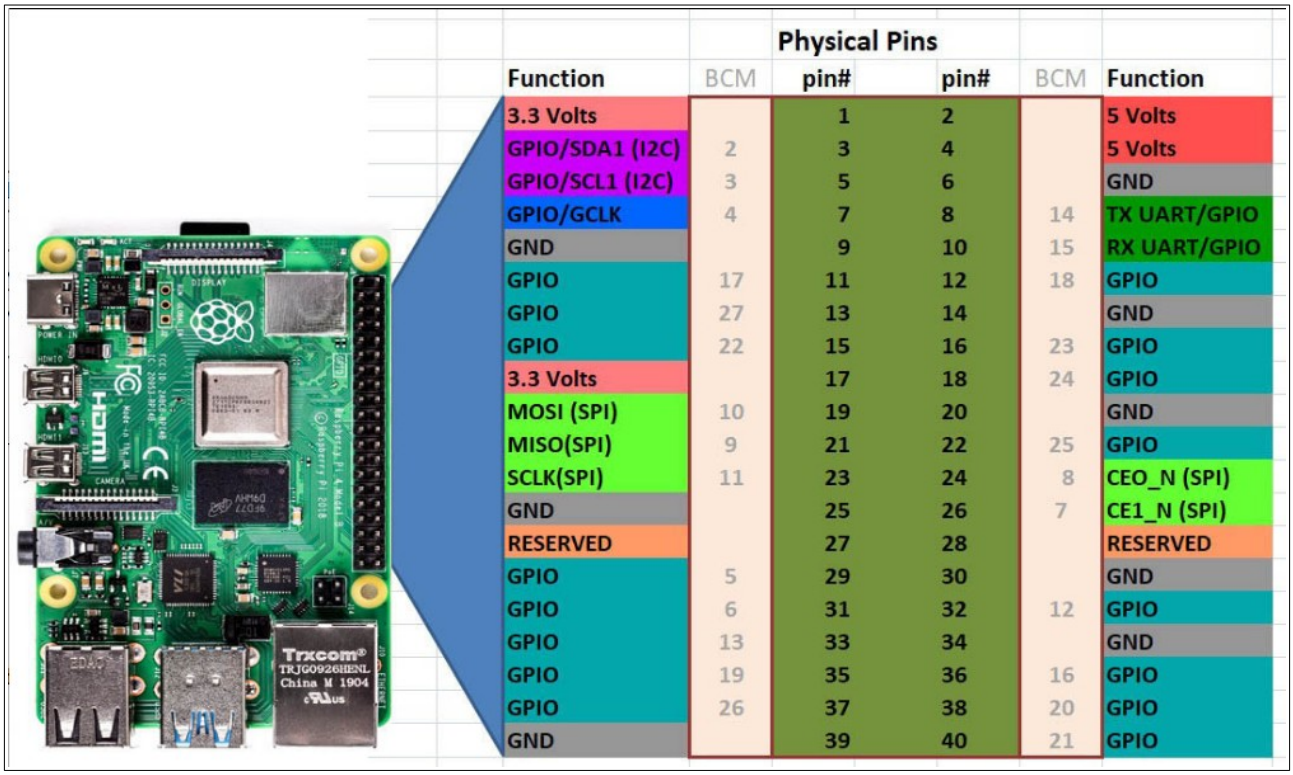
Ref:

- I2C protocol basics- 'How to use I2C in STM32F103C8T6? STM32 I2C Tutorial'
- Linux Kernel and Driver Development, Bootlin (aka Free electrons): linux-kernel-and-driver-dev.pdf : pg 172 – 185
- *Kernel doc : Implementing I2C device drivers*
- *Kernel doc: Implementing I2C client drivers in user-space*
- *Raspberry Pi SPI and I2C Tutorial*
- *Interfacing with I2C Devices (eLinux)*

With the DHT11 sensor chip
(Kernel drv: drivers/iio/humidity/dht11.c)

For all code / docs refer the GitHub repo here:
https://github.com/kaiwan/labrat_drv/tree/main/dht2x_temp_humd_i2c_driver

Raspberry Pi 4B (and/or R Pi 3 / R Pi 0W) pinout:



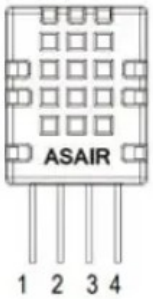
		Physical Pins			
Function	BCM	pin#	pin#	BCM	Function
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

DHT2x pinout

ref: https://proto-pic.co.uk/product/humidity-and-temperature-sensor-dht20/

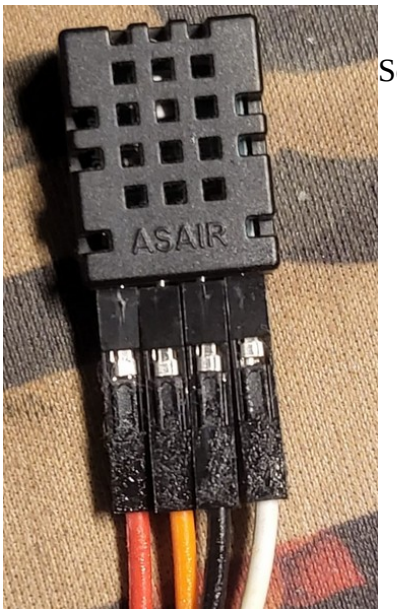
DHT20 Pin Out:

Pins	Name	Describe
1	VDD	Power supply(2.2v to 5.5v)
2	SDA	Serial Data Bidirectional port
3	GND	Ground
4	SCL	Serial clock Bidirectional port



VDD (power) to +3.3V (pin 1 on the Raspberry Pi)

<<
My DH120 wiring:



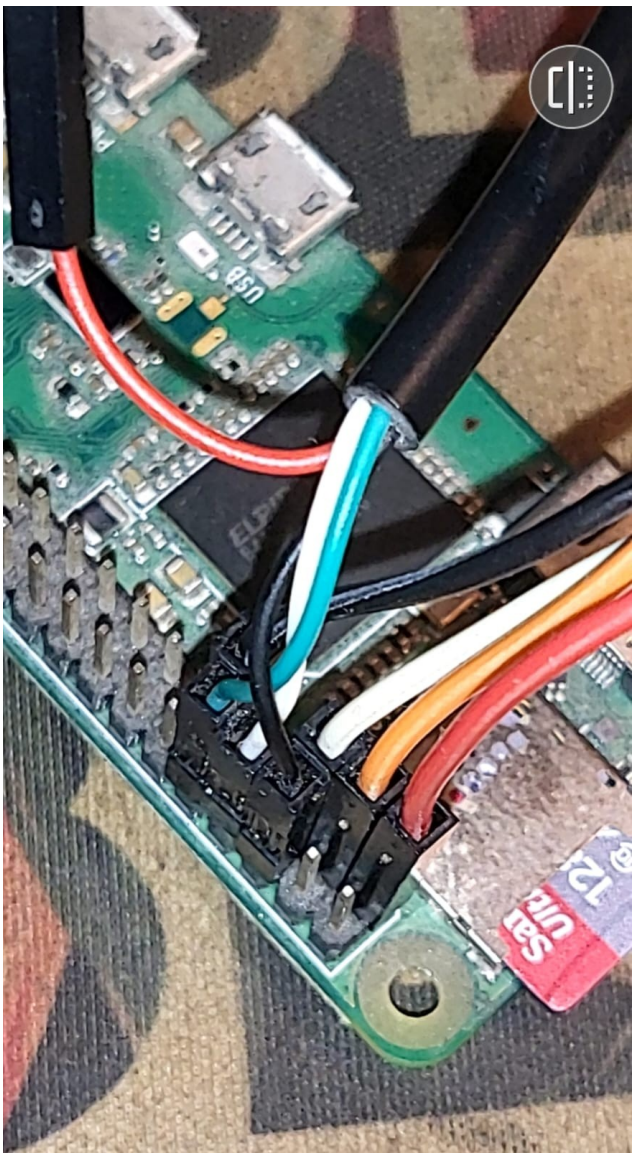
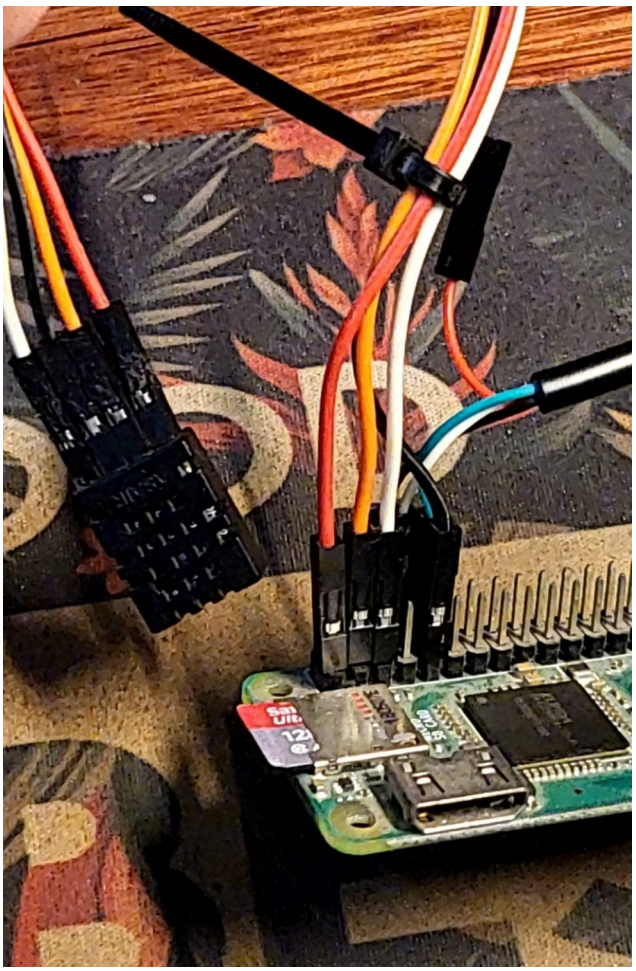
So, as per the datasheet:

- Pin 1: VDD : red wire
- Pin 2: SDA : orange wire
- Pin 3: GND : black wire
- Pin 4: SCL : white wire

>>

STEPS

1. Set up a connection – via a USB-to-serial console cable or SSH (preferred) – to your hardware board. Here, we assume it’s a Raspberry Pi (in particularly, am working with a Raspberry Pi 0W or the Pi 4 Model B; relevant for the DTS / DTB!)

	
Closeup: the USB-to-serial console cable's Black, White, Green (right-to-left) connectors in the GND, TX UART & RX UART GPIO pins (Raspberry Pi 0W). The DHT2x temperature/humidity sensor chip's wires can be behind it...	Closeup: the DHT2x temperature/humidity sensor chip's wires go into the appropriate GPIO connectors on the board in question (here, the Raspberry Pi 0W). The USB-to-serial console cable's Black, White, Green (right-to-left) connectors can be seen behind.

Login to the board.

2. Ensure that I2C is enabled (`sudo raspi-config`)

3. Ensure the following packages are installed:
   `sudo apt install -y i2c-tools libi2c-dev python3-smbus`

4. Obtain the board's DTS, edit it to include the DHT2x I2C sensor chip, compile it to the DT blob (DTB file), set the new DTB as the one used at boot.

1. Generate (reverse engineer!) the DTS :
   `dtc -I fs -O dts -@ /proc/device-tree/ > my_orig.dts`
   (-@ for symbols)

2. Modify it appropriately; save as a different file (f.e. `my_rpi4b_dht2x.dts`)
In the repo, here's the edited DTS..

<< How do we know which I2C bus it's on?
```
$ i2cdetect -l
i2c-1 i2c         bcm2835 (i2c@7e804000)           I2C adapter
```
>>

The stanza added to the R Pi DT source:
```
...
i2c@7e804000 {
   pinctrl-names = "default";
   #address-cells = <0x01>;
   [...]

   /* KNB: added node for DHT2x temp/humd sensor chip, to match
     my driver
    * Also note it's added in the right place, under the relevant
     I2C node
    * How to know?
    * i2cdetect -l shows the I2C buses along with their address; so
    * i2c-1 is here (on the RPi0W):
    *    i2c-1   i2c             bcm2835         (i2c@7e804000) ...
    * As well, i2cdetect -y 1 shows this chip is detected at 0x38,
    * hence we know it's on bus 1.
    * So: I2C bus 1 is at 0x7e804000 which is what this node describes;
    * hence we put our 'new' device - the DHT22 chip - here as a child
    * of this bus.
    */
    dht2x: dht22@0 {
         compatible = "knb,dht2x_kdrv";
         reg = <0x38>;
         pinctrl-names = "default";
         status = "okay";
   };
};
[...]
```

5. Compile DTS:
```
$ dtc my_rpi4b_dht2x.dts -o my_rpi4b_dht2x.dtb 2>&1 |cut -d: -f2-|
grep -i dht2
256.19-263.6: Warning (i2c_bus_reg): /soc/i2c@7e804000/dht22@0: I2C
bus unit address format error, expected "38"
```

(grep for the Warning and) Looks like we can ignore this warning...

6. Copy the new DTB into /boot, then edit **/boot/config.txt** and add this line
`device_tree=my_rpi4b_dht2x.dtb`
in order to override the default DTB with ours...
   (*Ensure* you keep the original DTB (here, it's */boot/bcm2711-rpi-4-b.dtb*) intact!)
   - Ref:
     - Raspberry Pi /boot/config.txt :
       https://www.raspberrypi.com/documentation/computers/config_txt.html

- *Raspberry Pi DTBs, overlays and config.txt* : https://www.raspberrypi.com/documentation/computers/configuration.html#part3.1

Test by rebooting; if all okay, the board reboots correctly, and you can see your new entry for the DHT2x within /proc/device-tree !
NOTE- if your board does not reboot, it's likely because the DTB isn't good; remove the 'device_tree=<...>' line in config.txt (IOW, let it use the original DTB), reboot, fix the issues and retry...

Rebooted with the new DTB; can see it's ok:

```
$ ls /proc/device-tree/soc/i2c@7e804000/dht22@0/
'#clock-cells'   compatible   name   pinctrl-names   reg   status

$ cat /proc/device-tree/soc/i2c@7e804000/dht22@0/*
knb,dht2x_kdrvdht228okaydht2x_temp_humd_i2c_driver $
```

7. Shutdown, power off (if not already done, connect the DHT2x sensor to your board).

8. After starting up with it attached, i2cdetect should show it:

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- 38 -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Perfect – it's on I2C bus 1, device address 0x38, as expected!

<<
When this fails to occur, check:

- Is the device wired correctly?
  - Recheck the pinout very carefully
  - Are the pins oriented correctly as per the pinout diagram?
  - Are the wires making good contact?
- If the I2C address slot shows 'UU', an existing kernel driver has 'taken over' the chip and is driving it
  - modify the Device Tree (DT) so as to claim it ourselves! (covered shortly)
- If nothing shows up, do we need to modify / add a node to the DT ? (this is the case on the Beagleboard's for instance)

>>

9. Write / edit the device driver, build it, test, install it:

(I found that without the `sudo depmod` in the Makefile, the driver gets installed but not detected and loaded up at boot, even with putting it into /etc/modules-load.d/<name>.conf !

```
rpi 5.15.76-v8+ # grep -i dht2x *
grep: build: Is a directory
grep: extra: Is a directory
grep: kernel: Is a directory

rpi 5.15.76-v8+ # depmod
[...]
rpi 5.15.76-v8+ # grep -i dht2x *
grep: build: Is a directory
grep: extra: Is a directory
grep: kernel: Is a directory
modules.alias:alias i2c:knb,dht2x dht2x_kdrv
modules.alias:alias of:N*T*Cknb,dht2xC* dht2x_kdrv
modules.alias:alias of:N*T*Cknb,dht2x dht2x_kdrv
grep: modules.alias.bin: binary file matches
modules.dep:extra/dht2x_kdrv.ko.xz:
grep: modules.dep.bin: binary file matches
rpi 5.15.76-v8+ #
)
```

The driver code and Makefile is *in the GitHub repo*...

[Though unnecessary here, in general, to have the kernel auto-load the driver, create this file:

```
$ cat /etc/modules-load.d/dht2x_kdrv.conf
# The DHT2x temp+humd I2C sensor chip
dht2x_kdrv
$
```

Here, the kernel (I2C) bus driver, detecting that the DHT2x chip's present, auto-loads (via the udev mechanism) the driver! This is as we updated the Device Tree to reflect that the chip's present...

10. Reboot; the driver should now be auto-loaded; the (I2C) bus driver pairs it with the sensor chip and the probe() method gets called. Great!

```
$ lsmod |grep dht
dht2x_kdrv             16384  0
$ dmesg |grep -i dht2x
[    3.199212] dht2x_kdrv: loading out-of-tree module taints kernel.
[    5.017455] dht2x 1-0038: dht2x_probe(): hey, in probe! name=dht2x addr=0x38
[    5.349181] dht2x 1-0038: dht2x_probe(): chip status (0x1c): calibration[b3]:
0x8   busy[b7]: 0x0
[    5.349218] dht2x 1-0038: dht2x_probe(): chip found
$
```

Moreover, the sysfs hooks are setup and ready to use; a quick demo shows its working just fine:

```
$ ls -l /sys/bus/i2c/devices/1-0038/
total 0
-r--r--r-- 1 root root 4096 Nov 25 14:28 dht2x_humd
-r--r--r-- 1 root root 4096 Nov 25 14:28 dht2x_temp
lrwxrwxrwx 1 root root    0 Nov 25 14:01 driver ->
../../../../../../bus/i2c/drivers/dht2x/
-r--r--r-- 1 root root 4096 Nov 25 14:28 modalias
-r--r--r-- 1 root root 4096 Nov 25 14:01 name
lrwxrwxrwx 1 root root    0 Nov 25 14:28 of_node ->
'../../../../../../firmware/devicetree/base/soc/i2c@7e804000/dht22@0'/
drwxr-xr-x 2 root root    0 Nov 25 14:28 power/
lrwxrwxrwx 1 root root    0 Nov 25 14:01 subsystem -> ../../../../../../bus/i2c/
```

```
-rw-r--r-- 1 root root 4096 Nov 25 14:01 uevent
$

$ cat /sys/bus/i2c/devices/1-0038/dht2x_humd
77507$
$ cat /sys/bus/i2c/devices/1-0038/dht2x_temp
23427$
$
```

Realize, of course, that the

- humidity value is in milli-percentage points (so humidity is currently 77.507%)
- similarly, temperature is expressed in millidegrees Celsius (so temperature is currently 23.427 degC)

Success!
