

# I2C and the DHT2x temperature + humidity sensor

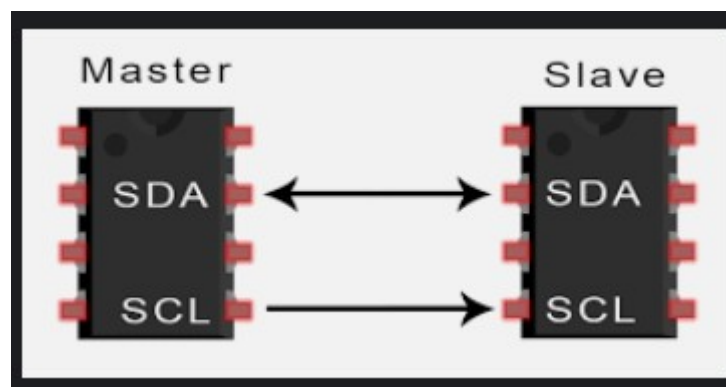
First, learn a little about the very popular widely used I2C (Inter-Inter Connect) 2-wire protocol that drives chip like this one!

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>



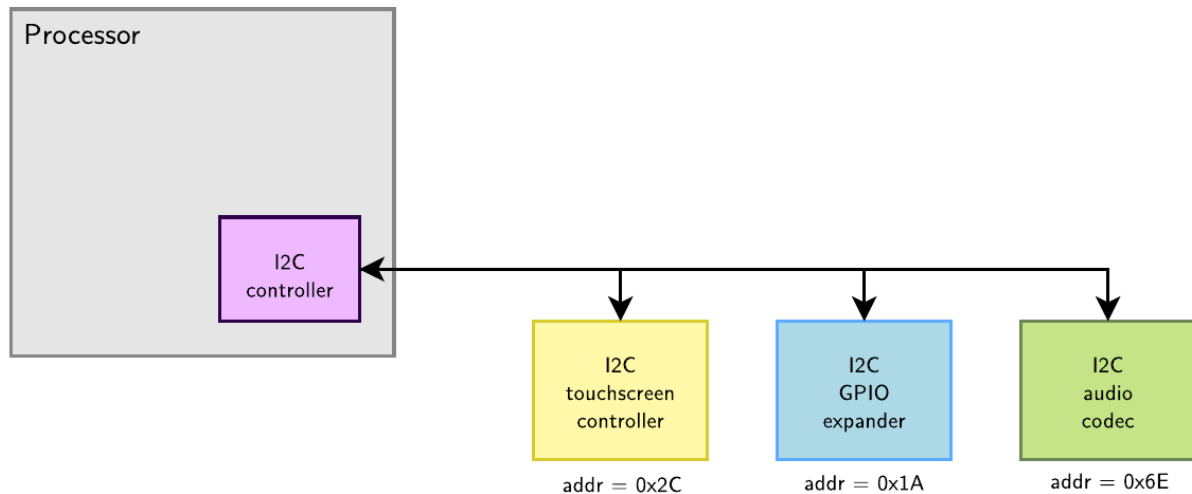
<Master/Controller>

<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.’



## An I2C bus example



bootlin - Kernel, drivers and embedded Linux - Development, consulting, training and support - <https://bootlin.com>

### 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:


- [‘How I2C Communication Works’, LinkedIn post by Housseem Akermi, Jan 2025](#)
- 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:

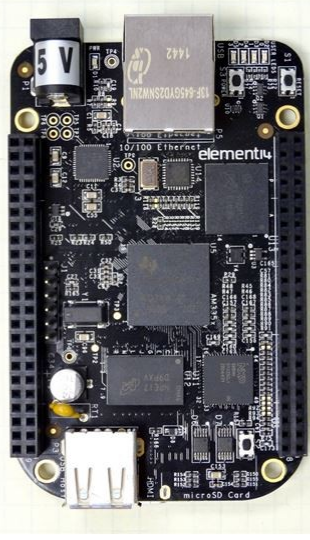
[https://github.com/kaiwan/labrat\\_drv/tree/main/dht2x\\_temp\\_humd\\_i2c\\_driver](https://github.com/kaiwan/labrat_drv/tree/main/dht2x_temp_humd_i2c_driver)

## Raspberry Pi 4B (and/or R Pi 3 / R Pi 0W) Pinout

	Function	BCM	Physical Pins		BCM	Function
			pin#	pin#		
	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

## TI BeagleBone Black (BBB) Pinout (Aarch32) Pinout

### Beaglebone Black Pinout Diagram

P9					P8			
Function	Physical Pins		Function		Function	Physical Pins		Function
DGND	1	2	DGND		DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V		MMC1_DAT6	3	4	MMC1_DAT7
VDD 5V	5	6	VDD 5V		MMC1_DAT2	5	6	MMC1_DAT3
SYS 5V	7	8	SYS 5V		GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESET		GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60		GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	EHRPWM1A		EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B		GPIO_47	15	16	GPIO_46
SPIO_CSO	17	18	SPIO_D1		GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C_SDA		EHRPWM2A	19	20	MMC1_CMD
SPIO_DO	21	22	SPIO_SLCK		MMC1_CLK	21	22	MMC1_DAT5
GPIO_49	23	24	UART1_TXD		MMC1_DAT4	23	24	MMC1_DAT1
GPIO_117	25	26	UART1_RXD		MMC1_DAT0	25	26	GPIO_61
GPIO_115	27	28	SP11_CSO		LCD_VSYNC	27	28	LCD_PCLK
SP11_DO	29	30	GPIO_112		LCD_HSYNC	29	30	LCD_AC_BIAS
SP11_SLCK	31	32	VDD_ADC		LCD_DATA14	31	32	LCD_DATA15
AIN4	33	34	GND_ADC		LCD_DATA13	33	34	LCD_DATA11
AIN6	35	36	AIN5		LCD_DATA12	35	36	LCD_DATA10
AIN2	37	38	AIN3		LCD_DATA8	37	38	LCD_DATA9
AIN0	39	40	AIN1		LCD_DATA6	39	40	LCD_DATA7
GPIO_20	41	42	ECAPWMO		LCD_DATA4	41	42	LCD_DATA5
DGND	43	44	DGND		LCD_DATA2	43	44	LCD_DATA3
DGND	45	46	DGND		LCD_DATA0	45	46	LCD_DATA1

**LEGEND**

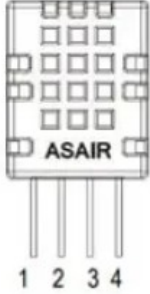
- Power, Ground, Reset
- Digital Pins
- PWM Output
- 1.8 Volt Analog Inputs
- Shared I2C Bus
- Reconfigurable Digital

## DHT2x pinout

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

### DHT20 Pin Out:

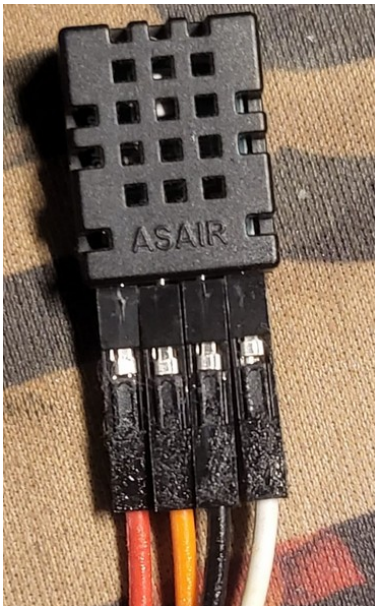
Pins	Name	Describe
1	VDD	Power supply(2.2v to 5.5v)
2	SDA	Serial Data Bidirectional port
3	GND	Ground
4	SCL	Serial clock Bidirectional port



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

<<

My DH120 wiring:



So, as per the datasheet:

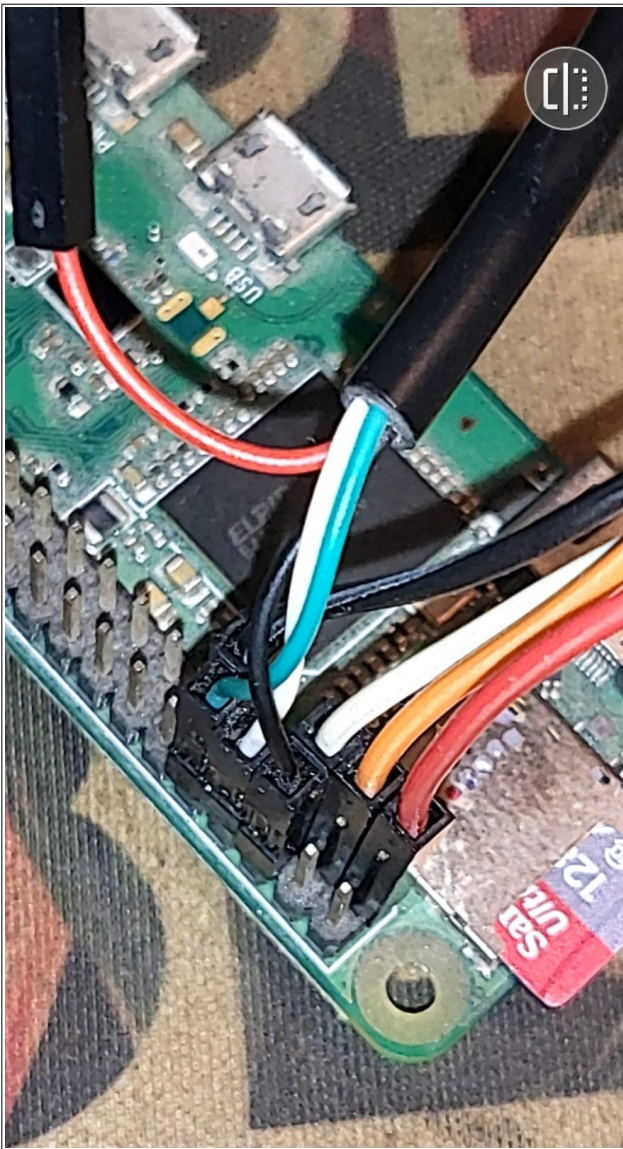
- Pin 1: VDD : red wire
- Pin 2: SDA : orange wire
- Pin 3: GND : black wire
- Pin 4: SCL : white wire

>>

## STEPS

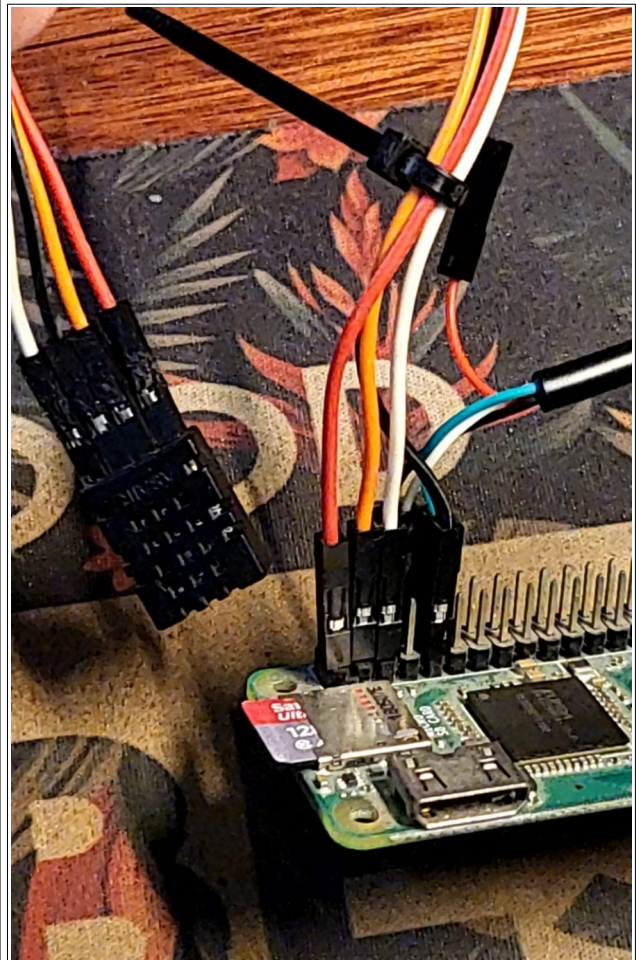
1. Set up a connection – via a USB-to-serial console cable or SSH (preferred) – to your hardware board. Here, we assume it's a Raspberry Pi (in particularly, am working with a Raspberry Pi 0W or the Pi 4 Model B; relevant for the DTS / DTB!)





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...

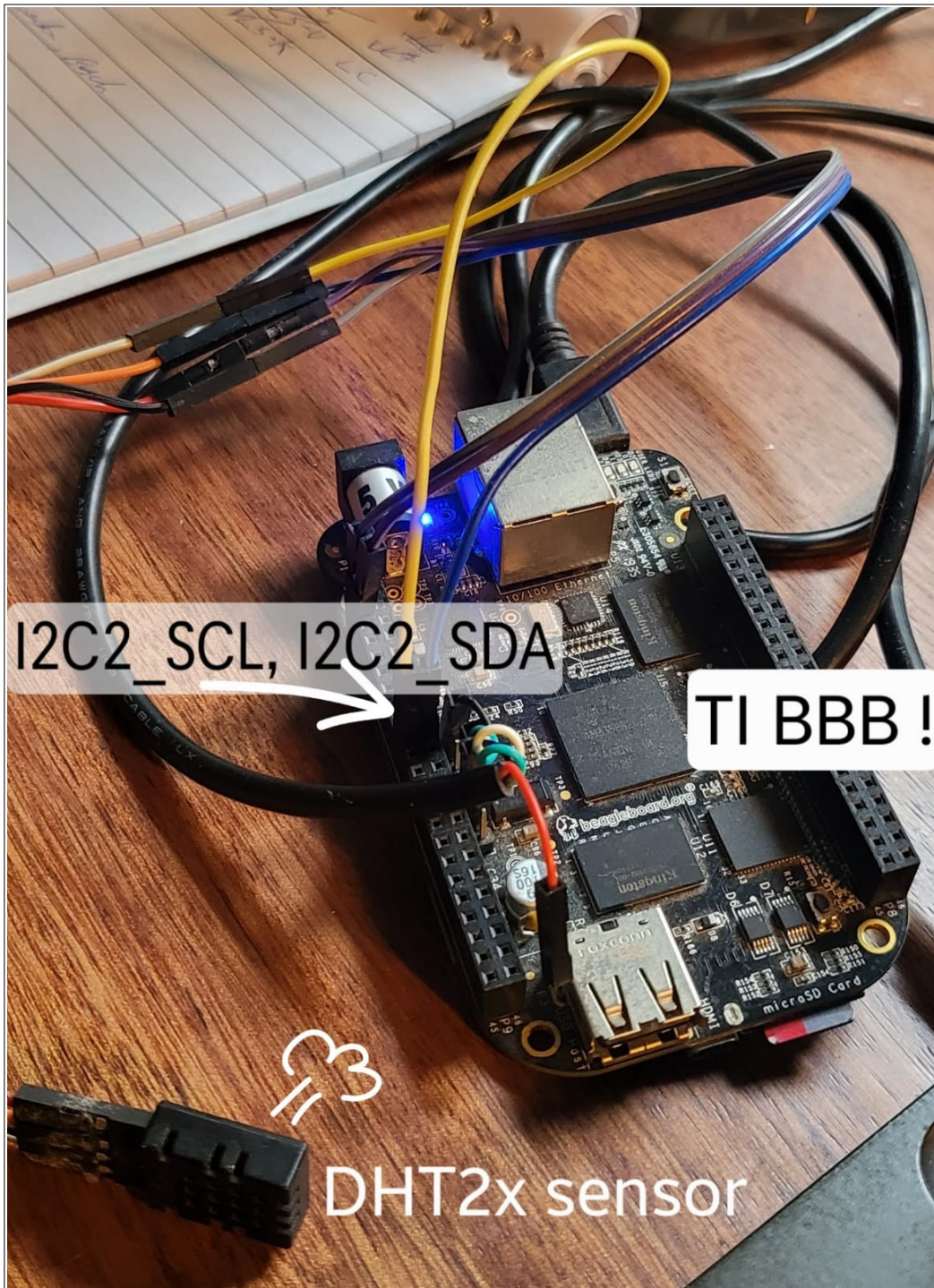


Closeup: the DHT2x temperature/humidity sensor chip's wires go into the appropriate GPIO connectors on the board in question (here, the Raspberry Pi 0W).

The USB-to-serial console cable's Black, White, Green (right-to-left) connectors can be seen behind.



## On the TI BBB



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 = "knbc,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](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_kdrv dht228okay dht2x_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:knbc,dht2x dht2x_kdrv
modules.alias:alias of:N*T*Cknbc,dht2xC* dht2x_kdrv
modules.alias:alias of:N*T*Cknbc,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!


---

# Deploying the same DHT2x driver on the TI BeagleBone Black (BBB)

Connections on the P9 header (see photo on following page):

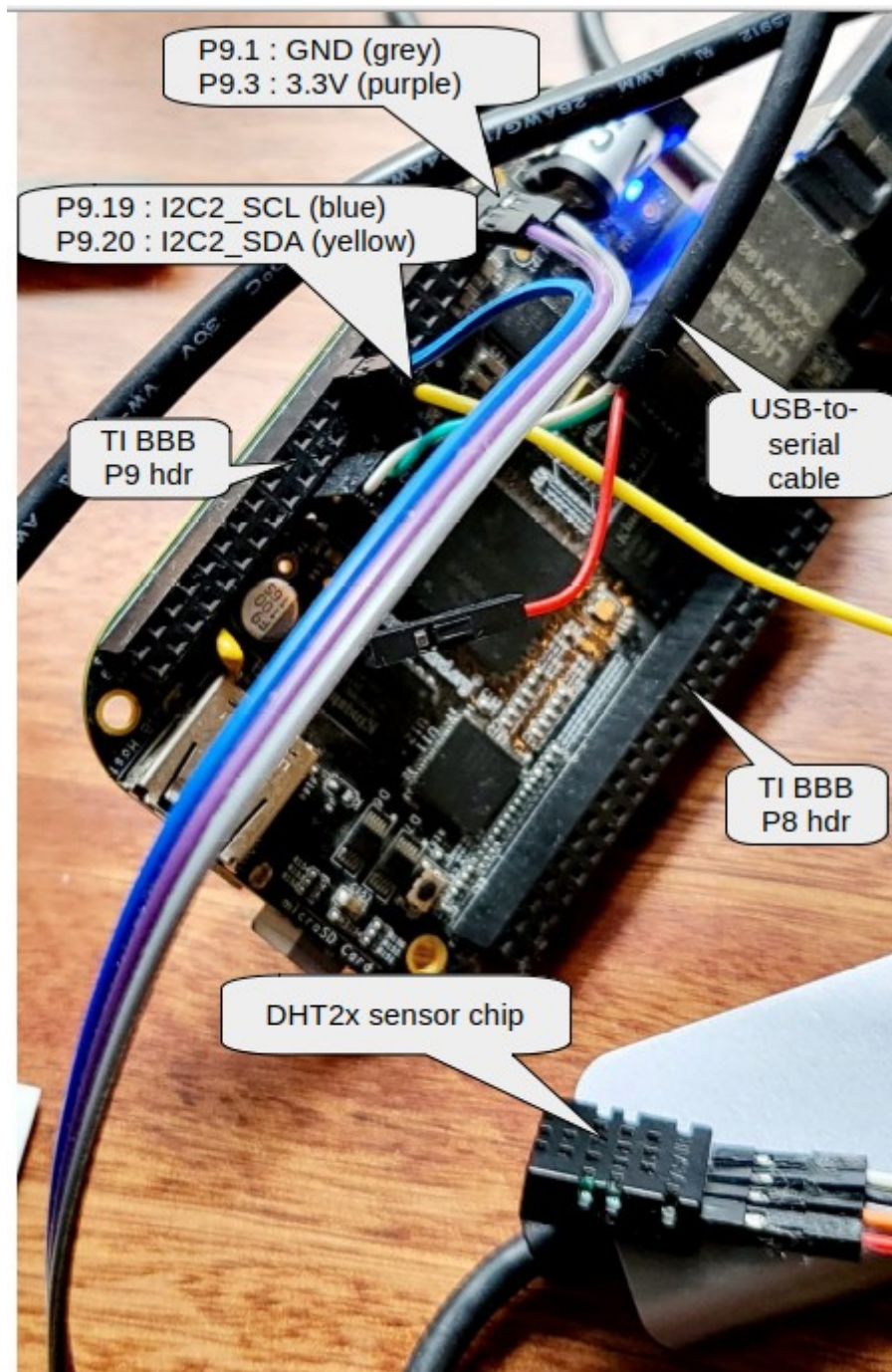
Physical pin #	Function	DHT2x connection
1	DGND	Connect DHT2x GND wire
3	VDD 3.3V	Connect DHT2x VCC wire
19	I2C2_SCL	Connect DHT2x SCL (serial clock) wire
20	I2C2_SDA	Connect DHT2x SDA (serial data) wire

## Beaglebone Black Pinout Diagram

P9					P8			
Function	Physical Pins		Function		Function	Physical Pins		Function
DGND	1	2	DGND		DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V		MMC1_DAT6	3	4	MMC1_DAT7
VDD 5V	5	6	VDD 5V		MMC1_DAT2	5	6	MMC1_DAT3
SYS 5V	7	8	SYS 5V		GPIO_66	7	8	GPIO_67
PWR_BTN	9	10	SYS_RESET		GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60		GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	EHRPWM1A		EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B		GPIO_47	15	16	GPIO_46
SPI0_CS0	17	18	SPI0_D1		GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C_SDA		EHRPWM2A	19	20	MMC1_CMD
SPI0_DO	21	22	SPI0_SLCK		MMC1_CLK	21	22	MMC1_DAT5
GPIO_49	23	24	UART1_TXD		MMC1_DAT4	23	24	MMC1_DAT1
GPIO_117	25	26	UART1_RXD		MMC1_DAT0	25	26	GPIO_61
GPIO_115	27	28	SP11_CS0		LCD_VSYNC	27	28	LCD_PCLK
SP11_DO	29	30	GPIO_112		LCD_HSYNC	29	30	LCD_AC_BIAS
SP11_SCLK	31	32	VDD_ADC		LCD_DATA14	31	32	LCD_DATA15
AIN4	33	34	GND_ADC	<b>LEGEND</b> Power, Ground, Reset Digital Pins PWM Output 1.8 Volt Analog Inputs Shared I2C Bus Reconfigurable Digital	LCD_DATA13	33	34	LCD_DATA11
AIN6	35	36	AIN5		LCD_DATA12	35	36	LCD_DATA10
AIN2	37	38	AIN3		LCD_DATA8	37	38	LCD_DATA9
AIN0	39	40	AIN1		LCD_DATA6	39	40	LCD_DATA7
GPIO_20	41	42	ECAPWMO		LCD_DATA4	41	42	LCD_DATA5
DGND	43	44	DGND		LCD_DATA2	43	44	LCD_DATA3
DGND	45	46	DGND		LCD_DATA0	45	46	LCD_DATA1

<Note: we're using the I2C2 bus (not I2C1) of the BBB>





[Source repo](#)

The DT Overlay ([here](#)):

```
bbb dht2x_temp_humd_i2c_driver $ cat bbb_dts/bbb_dht2x.dts
/*
 * DTB0 - DT overlay - for the DHT2X temperature+humidity sensor chip on the TI
BBB
 * Inspired by - on the default Debian OS BBB rootfs :
 * /opt/source/dtb-5.10-ti/src/arm/overlays/BB-I2C2-BME680.dts
 */
/dts-v1/;
/plugin/;
```

```

/*
 * Helper to show loaded overlays under: /proc/device-tree/chosen/overlays/
 */
&{/chosen} {
    overlays {
        BB-I2C2-DHT2X.kernel = "Tue Oct 14 22:31:00 2000";
    };
};

&i2c2 {
    status = "okay";

    clock-frequency = <100000>;

    #address-cells = <1>;
    #size-cells = <0>;

    dht22@38 {
        compatible = "asair,dht2x_kdrv";
        reg = <0x38>;
        status = "okay";
    };
};

```

### *Compile driver and DTBO*

**\$ make**

```

...
LD [M]
/home/debian/kaiwanTECH/labrat_drv/dht2x_temp_humd_i2c_driver/dht2x_kdrv.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.10.168-ti-r71'
if [ "y" = "y" ]; then \
    strip --strip-debug dht2x_kdrv.ko ; \
fi
--- compile the Device Tree Blob (DTB) Overlay from the DTS and cp to
/boot/.../overlays ---
dtc -@ -I dts -O dtb -o bbb_dts/BBB-DHT2X.dtbo bbb_dts/bbb_dht2x.dts
sudo cp bbb_dts/BBB-DHT2X.dtbo /boot/dtbs/5.10.168-ti-r71/overlays
Add the following line to /boot/uEnv.txt (if not already there):
dtb_overlay=/boot/dtbs/5.10.168-ti-r71/overlays/BBB-DHT2X.dtbo
...

```

The ‘make’ copies the DT overlay into the appropriate folder – either /lib/firmware or /boot/dtbs/5.10.168-ti-r71/overlays/ (the latter here)..

Edit /boot/uEnv.txt adding this line so that the new DT overlay is recognized and loaded by U-Boot at boot (as shown above)..

Reboot...

```

...
U-Boot 2022.04-ge0d31da5 (Aug 04 2023 - 18:48:26 +0000)

CPU   : AM335X-GP rev 2.1
Model : TI AM335x BeagleBone Black
DRAM  : 512 MiB
...
uboot_overlays: [uboot_base_dtb=am335x-boneblack-uboot-univ.dtb] ...
uboot_overlays: Switching too: dtb=am335x-boneblack-uboot-univ.dtb ...

```

```

loading /boot/dtbs/5.10.168-ti-r71/am335x-boneblack-uboot-univ.dtb ...
210757 bytes read in 21 ms (9.6 MiB/s)
Found 0 extension board(s).
uboot_overlays: [fdt_buffer=0x60000] ...
uboot_overlays: loading /boot/dtbs/5.10.168-ti-r71/overlays/BB-ADC-00A0.dtbo ...
645 bytes read in 7 ms (89.8 KiB/s)
uboot_overlays: loading /boot/dtbs/5.10.168-ti-r71/overlays/BB-BONE-eMMC1-01-
00A0.dtbo ...
1605 bytes read in 7 ms (223.6 KiB/s)
uboot_overlays: loading /boot/dtbs/5.10.168-ti-r71/overlays/BB-HDMI-TDA998x-
00A0.dtbo ...
5321 bytes read in 6 ms (865.2 KiB/s)
uboot_overlays: [dtb_overlay=/boot/dtbs/5.10.168-ti-r71/overlays/BBB-DHT2X.dtbo]
...
uboot_overlays: loading /boot/dtbs/5.10.168-ti-r71/overlays/BBB-DHT2X.dtbo ...
570 bytes read in 7 ms (79.1 KiB/s)
loading /boot/initrd.img-5.10.168-ti-r71 ...
...

```

<< FYI:

“There is also a useful tool called `show-pins` which will show you the state of the current PinMux settings. The `show-pins` tool is a clever Perl script which compiles information from various sources (including `config-pin`) to show you the current state of the PinMux in a nice format. You can obtain the `show-pins` tool from <https://github.com/mvduin/bbb-pin-utils>.”

>>

Verify the pinmux:

```

$ show-pins | grep -E "P9.19|P9.20"
P9.20 / cape i2c sda          94 fast rx  up  3 i2c 2 sda
ocp/P9_20_pinmux (pinmux_P9_20_default_pin)
P9.19 / cape i2c scl          95 fast rx  up  3 i2c 2 scl
ocp/P9_19_pinmux (pinmux_P9_19_default_pin)

```

Good, it's as expected; now lets load the driver and run the script to show temperature and humidity values once/second:

```

bbb dht2x_temp_humd_i2c_driver $ sudo insmod ./dht2x_kdrv.ko
bbb dht2x_temp_humd_i2c_driver $ ./disp_temp_humd.sh
./disp_temp_humd.sh: line 12: warning: command substitution: ignored null byte
in input
Detected we're running on the TI AM335x BeagleBone Black
temperature(milliC),rel_humidity(milli%)
26756,72674
26757,72668
26757,72657
...

```

Done.

---