RHT03 Sensor How-To Guide

This guide builds upon a pre-existing guide included in submission. Previous guides also at:
CMPT433 Course Website -> Resources -> Student CMPT433 How-To Guides

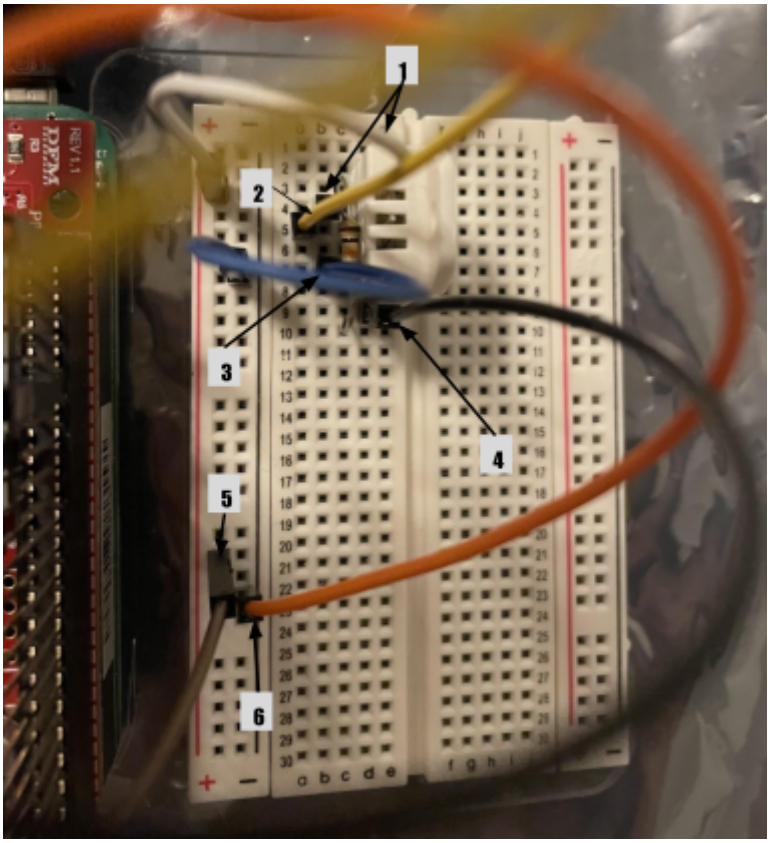
Wiring

Wires and resistor(s) needed:

- 3 male to male wires
- 3 female to male wires
- 1 1K resistor (borrowed from professor)

Step 1: Follow *part 2. Connecting the Breadboard* in the *RHT03 Sensor How To Guide*[1] to set up the wiring.

Note: - We used a 1K resistor as suggested in the sensor datasheet instead of a 10K resistor used in the above student guide.



* See simplified Hardware Wiring Diagram on page 3

White wire (wire 1):
- Sensor pin 1 (power)
- To 3.3V pin on BBG

Yellow wire (wire 2):
- Sensor pin 2 (data)
- To p8_15 on BBG

Blue wire (wire 3):
- Sensor pin 4 (ground)
- To ground on BBG

Black wire (wire 4):
- One end of the resistor
- To p8_12 on BBG

Brown wire (wire 5):
- p9_7 on BBG

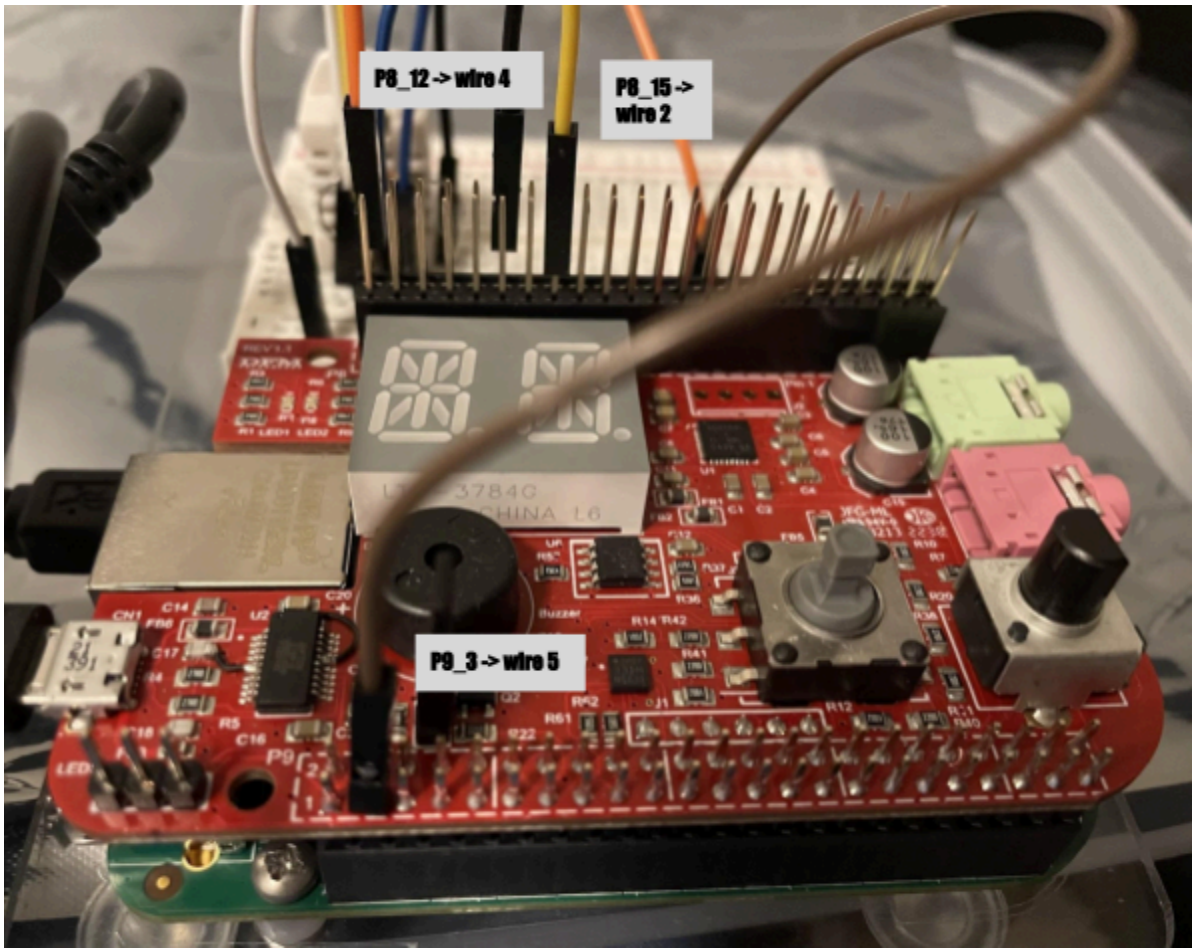
Orange wire (wire 6):
- Connect to ground
- p8_1 or p8_2 on BBG

Resistor:
- Pin 2 (data)
- To black wire (wire 4)

[1] RHT03 Sensor How To Guide:
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2016-student-howtos/RHT03Sensor-HowtoGuide.pdf

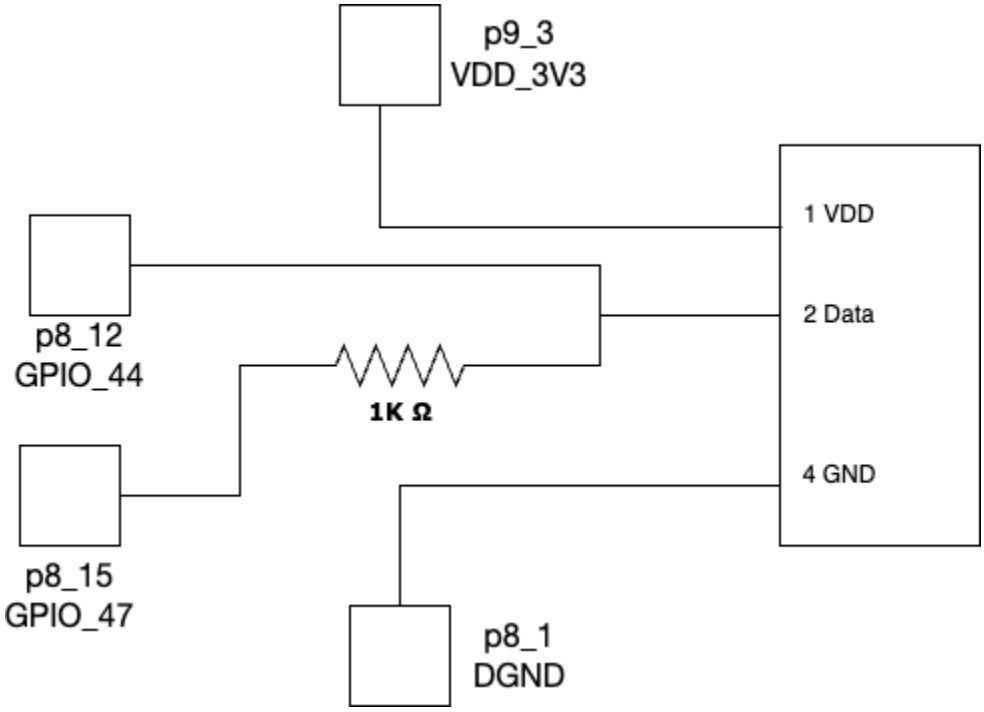
Step 2: Connect to the beaglebone

- Connect the sensor data pin (pin 2) to p8_15 on the beaglebone, and
- Ensure, one end of the 1K resistor is connected to the sensor data pin (as shown above), now connect the other end of the resistor to p8_12 on the beaglebone.
  Note:
  - In the PRU code, p8_15 will need to be configured as 'pruin', and p8_12 configured as 'pruout'.
  - Check the P8 Header[2], P9 Header[3] files provided on the course website to choose the proper pins as PRU input and output pins for your BeagleBone.
  - Consider replacing p8_15 with another pin if Joystick is used in the PRU code.



[2] P8 header: https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/guides/files/bbg_docs/BeagleboneBlackP8HeaderTable.pdf
[3] P9 header: https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/guides/files/bbg_docs/BeagleboneBlackP9HeaderTable.pdf

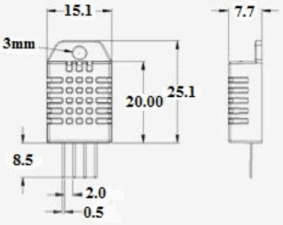
Hardware Wiring Diagram:



Brief explanation of how the sensor sends data

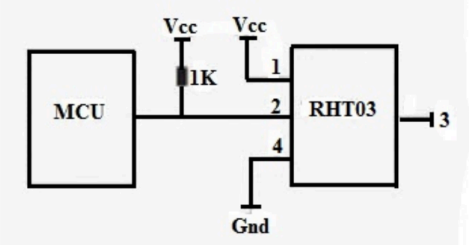
Diagram from the sensor datasheet[4]

4. Dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD—power supply
2	DATA--signal
3	NULL
4	GND



[4] Digital relative humidity & temperature sensor RHT03 Datasheet: https://cdn.sparkfun.com/datasheets/Sensors/Weather/RHT03.pdf?_gl=1*6pnv6o*_ga*MTkwNDY1OTAxMy4xNzA5NTcyNTYx*_ga_T369JS7J9N*MTcwOTU3MjU2MC4xLjEuMTcwOTU3MzIyMy41MS4wLjA.

General logic of the data transmission cycle

The below picture briefly shows the interaction between the host and the sensor, and converts step 1) and step 2) on the RHT03 datasheet (page 3 and 4) into pseudocode.

```
Initially the bus status is high voltage level

// because one start signal for one response data from the sensor
loop
{
    The host sends start signal:
        (1) pulls low for 1 ~ 10 ms
        (2) pulls up for 20 ~ 40 us

    The sensor sends response signal and prepare to send data:
        (1) pulls low for 80 us
        (2) pulls up for 80 us

    The sensor starts sending data:
        loop for 40 times // because the data is consists of 40 bits and sent bit by bit
        {
            (1) pulls low for 50 us (every bit's transmission begins with this pull low)
            (2.a) if pulls up for 26 ~ 28 us  -> the bit sent is "0"
            (2.b) if pulls up for 70 us       -> the bit sent is "1"
            * Note: can use an arbitrary value such as '40 us' to decide whether the bit
            is "0" or "1"
        }
}
```

Note:

- To decide if the bit sent is "0" or "1", we need to measure how long the high-voltage-level is. One way is to use an arbitrary value between 28 and 70 as the threshold (such as 40 μs). Another way (as used by the 2022 Fall student group) is to wait for 40 μs, and check if the current voltage level is high or low.

Converting the 40-bit data to the temperature and humidity values

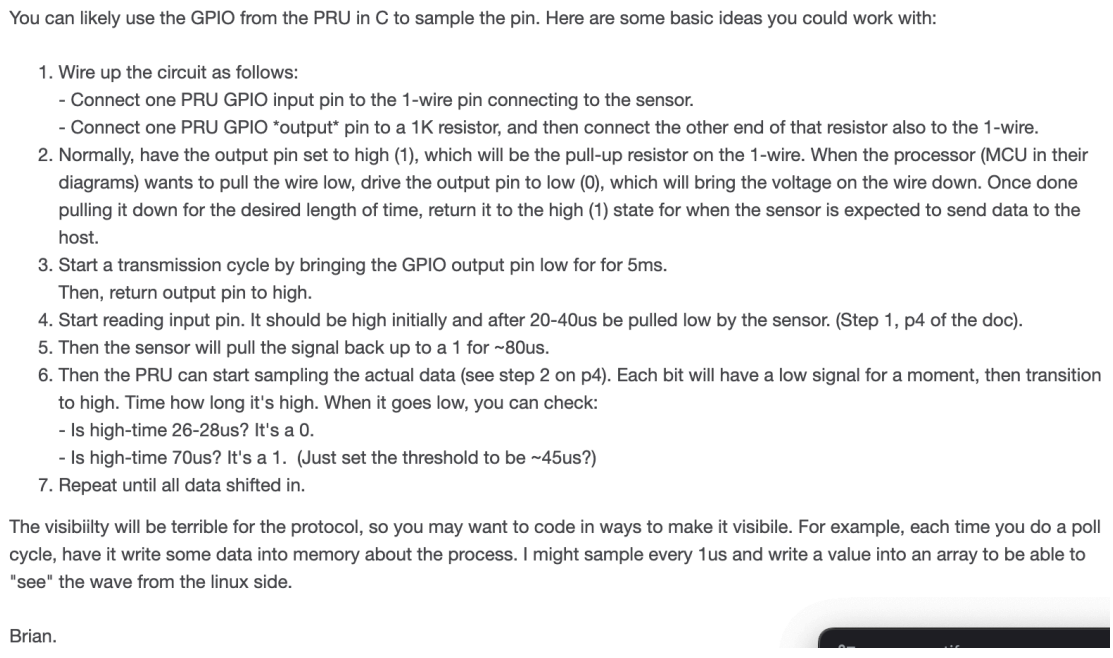
Please refer to the *beginReadingFromSensor()* function in the *tempSensor.c* file provided by 2022 Fall student group.

Experimenting with the sensor

- Get the data and convert to humidity and temperature values
    - Refer to How-To RHT03 Sensor using PRU guide support files[5], and run the demo program by following their instructions. The temperature and humidity values will be printed out approximately every 2 seconds. Try observing if the readings will change when the environment changes..

Troubleshooting and other

- It is suggested after setting the bits (pulling high or low) in PRU, to wait/delay a certain amount of time before proceeding.
- When starting experimentation with the sensor, one way to deal with the low-visibility of this 1-wire protocol is using a data structure such as a boolean array (e.g., of size 2k), which can be used to store the bits read from the PRU input.
    - An example of what the array will look like:
      During the sensor response phase, the data-bus's voltage level will be pulled low for 80 µs. This means that the array will store consecutive 80 zeros.
      After these zeros, because of the sensor pulling up, the values stored in the array will be 80 ones (ideally).
- Tips shared by Dr.Brian (let us know if you don't want the piazza answer to appear here)



You can likely use the GPIO from the PRU in C to sample the pin. Here are some basic ideas you could work with:

1. Wire up the circuit as follows:
   - Connect one PRU GPIO input pin to the 1-wire pin connecting to the sensor.
   - Connect one PRU GPIO *output* pin to a 1K resistor, and then connect the other end of that resistor also to the 1-wire.
2. Normally, have the output pin set to high (1), which will be the pull-up resistor on the 1-wire. When the processor (MCU in their diagrams) wants to pull the wire low, drive the output pin to low (0), which will bring the voltage on the wire down. Once done pulling it down for the desired length of time, return it to the high (1) state for when the sensor is expected to send data to the host.
3. Start a transmission cycle by bringing the GPIO output pin low for for 5ms.
   Then, return output pin to high.
4. Start reading input pin. It should be high initially and after 20-40us be pulled low by the sensor. (Step 1, p4 of the doc).
5. Then the sensor will pull the signal back up to a 1 for ~80us.
6. Then the PRU can start sampling the actual data (see step 2 on p4). Each bit will have a low signal for a moment, then transition to high. Time how long it's high. When it goes low, you can check:
   - Is high-time 26-28us? It's a 0.
   - Is high-time 70us? It's a 1. (Just set the threshold to be ~45us?)
7. Repeat until all data shifted in.

The visibiilty will be terrible for the protocol, so you may want to code in ways to make it visibile. For example, each time you do a poll cycle, have it write some data into memory about the process. I might sample every 1us and write a value into an array to be able to "see" the wave from the linux side.

Brian.

---

5 https://opencoursehub.cs.sfu.ca/bfraser/solutions/433/StudentHowToGuides/2022Files-Ensc351/rht-sensor-demo.zip

Side notes:

Motivations for the guide: Since some python libraries referred to by some previous student groups were deprecated and we faced some difficulties to get all the dependencies installed, we need to use PRU to get the digital signals. Fortunately, one previous student group used a similar strategy to retrieve and process the sensor data, and provided support files (their demo code). However, their wiring created extra complexity which was difficult to fit with all our other sensors. Considering the complexity of wiring various sensors up, we simplified the wiring and omitted the non-relevant (LED and button) debugging component present in the previous guide. Therefore, this guide can be used to set up the basic wiring needed to work with the RHT03 sensor.

Copyright permission: this guide can be posted online for future students to access.