# Environmental Monitoring With a Raspberry Pi

Madison Levagood[a], George Iakovidis[b]

August 2026

[a] *University of British Columbia, Department of Physics, Agricultural Road, Vancouver, Canada*
[b] *Brookhaven National Laboratory, Upton, NY, U.S.A.*

**Summary:** This document provides a guide to creating an environmental monitoring system on a Raspberry Pi. It details how to connect a sensor, and includes example readout code in Python and C++. Elementary proficiency with soldering, the Linux Command Line, and some basic programming skills are assumed.

# Contents

# 1 Introduction

This document details how to create a humidity, temperature, and dew point monitoring system, such as the one used in the SR1 racks at CERN. The sufficiency of this system for a given purpose is naturally limited by the sensor(s) used. The specific sensor used here is appropriate for any system between 0 and 100% relative humidity and $-40$ and $125°C$. Additionally, this sensor should only be used where the accuracy of $\pm 1.8\%$ relative humidity and $\pm 0.2°C$ is deemed acceptable[1].

## 1.1 Materials

- Sensor: HYT221.
- Computer: Raspberry Pi 4 Model B. There is considerable flexibility here regarding the model; the important feature is the GPIO pins.
- Four-conductor cable.
- Soldering iron and solder.
- Wire crimps and crimper.
- Wire stripper.
- DuPont connectors.
- **Optional:** A simple PCB that connects the HYT pins to the wires inside the cable. Any method of creating a stable electrical connection between these elements is likely sufficient.

# 2 The Hardware

The HYT has four pins: SCL, VCC, GND, and SDA (see Figure 1)[1]. These must be connected to the corresponding GPIO pins on the Raspberry Pi (Figure 2)[2]. The simplest way to do this is with a four-conductor cable. A basic PCB can be used to provide convenient connection between the HYT and the cable. To do this, remove a few centimeters of the outermost insulating layer of the cable. The four wires inside will all have their own insulation, which should hopefully be colour-coded. Strip a few millimeters of this inner insulation off all four wires, and solder the exposed wire to the PCB, taking care to note which wire connects to which pin on the HYT, which should also be soldered to the PCB (see Figure 3).

On the opposite end of the cable (the end which will plug into the Raspberry Pi), again remove a few centimeters of the outer insulation, and strip a few millimeters of the inner insulation for all four wires. Crimp the wires and insert them in the DuPont connectors, then, with the Raspberry Pi off, connect the DuPont connectors to the corresponding GPIO pins.

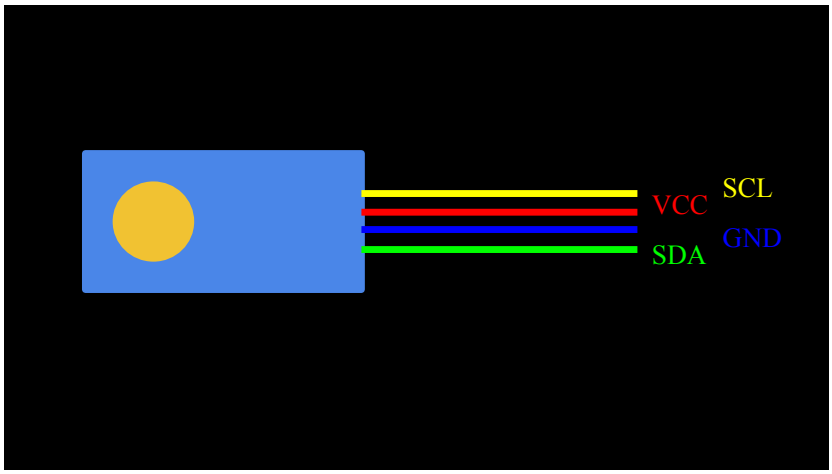


Figure 1: Schematic of an HYT221.

Note that the Raspberry Pi should never be unplugged without shutting it down first (see command below), as this can corrupt the SD card.

```
sudo shutdown -h now
```

If the Raspberry Pi does not yet have an operating system installed, see the Raspberry Pi documentation[3]. Plug the Raspberry Pi in, and SSH into it. Before writing any code, confirm that the Raspberry Pi detects the sensor by running

```
i2cdetect -y 1
```

The HYT221 defaults to $I^2C$ address 0x28, so the output should look like so:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- 28 -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

If the output looks correct, continue to Section 3.

# 3 The Software

The details of the software will naturally vary depending on what is to be done with the collected data. Sections 3.1 and 3.2 provide examples for how to readout the HYT in Python and C++, respectively. That readout could then go through an OPC UA server, for example, or get pushed to InfluxDB and

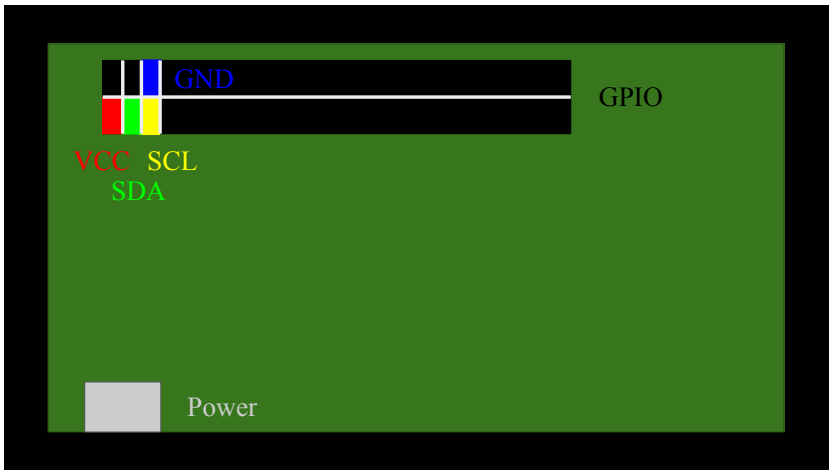


Figure 2: Schematic of the relevant GPIO pins on a Raspberry Pi, as viewed in the orientation where the silkscreen text is upright and the power cable plugs in at the bottom left corner.

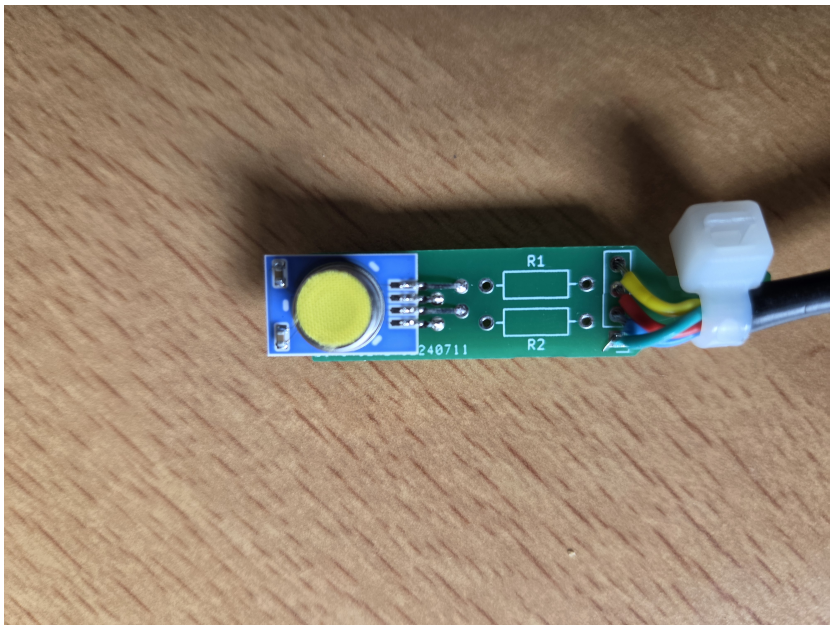


Figure 3: An example connection between the HYT221 and a four-conductor cable.

monitored with Grafana. In the coding examples, the data is simply printed to the Command Line.

The general method for reading the HYT data is to write via $I^2C$, wait, and then take a reading. Next, the raw data is converted to humidity and temperature values via bit-wise operations explained on page 17 of the HYT221 application note[1]. Finally, the humidity and temperature can be used to calculate a dew point, if desired. There are different ways to do this, but the formula used in this document is the following:

$$\alpha = \frac{AT}{B+T} + ln(H),$$

$$DewPoint = \frac{B\alpha}{A-\alpha},$$

Where $A = 17.27$, $B = 237.7°C$, $T$ is the temperature in Celsius, and $H$ is the relative humidity as a decimal (not a percent)[4].

## 3.1 Reading From the HYT in Python

```python
import numpy as np
from smbus2 import SMBus, i2c_msg
import time
from typing import List, Tuple

def read_i2c(i2c_bus: SMBus, i2c_address: int) -> List[int]:
    '''
    Read raw values over I2C.

    Parameters:
    i2c_bus    : The I2C bus from which to read.
    i2c_address: The I2C address.

    Returns:
    data: Raw data returned from HYT.
    '''

    #Write via I2C
    msg = i2c_msg.write(i2c_address, 4)
    i2c_bus.i2c_rdwr(msg)

    #Wait and read
    time.sleep(0.1)
    msg = i2c_msg.read(i2c_address, 4)
    i2c_bus.i2c_rdwr(msg)
    data = list(msg)

    return data

def convert_raw_data(data: List[int]) -> Tuple[float, float]:
    '''
    Convert raw data from HYT into humidity and temperature values.

    Parameters:
```

```
    data: list of values returned from raw HYT read.

    Returns:
    humidity   : The relative humidity percentage.
    temperature: The temperature in Celsius.
    '''

    humidity    = max(((data[0] & 0x3f) << 8 | data[1]) * (100.0 / 0x3fff), 0.0)
    temperature = (data[2] << 8 | (data[3] & 0xfc)) * (165.0 / 0xfffc) - 40.0

    return humidity, temperature

def calculate_dew_point(humidity: float, temperature: float) -> float:
    '''
    Calculate the dew point based on humidity and temperature.

    Parameters:
    humidity   : The relative humidity percentage.
    temperature: The temperature in Celsius.

    Returns:
    dew_point: The dew point in Celsius.
    '''
    a         = 17.27
    b         = 237.7
    alpha     = (a * temperature) / (b + temperature) + np.log(humidity / 100)
    dew_point = (b * alpha) / (a - alpha)

    return dew_point

def main() -> None:
    '''
    Read out temperataure and humdity from HYT221, calculate dew point, and print data to terminal.
    '''

    i2c_bus     = SMBus(1)
    i2c_address = 0x28

    data                  = read_i2c(i2c_bus, i2c_address)
    humidity, temperature = convert_raw_data(data)
    dew_point             = calculate_dew_point(humidity, temperature)

    print(f"Humidity: {humidity:.3}%.\nTemperature: {temperature:.3}C.\nDew Point: {dew_point:.3}C.")

if __name__ == '__main__':
    main()
```

### 3.2 Reading From the HYT in C++

Note that there are a few ways to do this in C++. The library used here, WiringPi[5], is convenient and beginner-friendly, but it is not the only option.

```cpp
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <iostream>
#include <unistd.h>
#include <cmath>

int main()
{
  int i2c_address = 0x28;

  //Write via I2C
  int wpi_setup = wiringPiI2CSetup(i2c_address);
  write(wpi_setup, "", 0);

  //Wait and read
  usleep(100000);
  unsigned char data[4];
  read(wpi_setup, data, 4);

  //Convert raw data to temperature and humidity
  double humidity    = ((data[0] & 0x3f) << 8 | data[1]) * (100.0 / 0x3fff);
  double temperature = (data[2] << 8 | (data[3] & 0xfc)) * (165.0 / 0xfffc) - 40.0;

  //Calculate dew point
  double a         = 17.27;
  double b         = 237.7;
  double alpha     = (a * temperature) / (b + temperature) + log(humidity / 100);
  double dew_point = (b * alpha) / (a - alpha);

  //Print results to terminal
  std::cout << "Humidity: " << humidity << "%\n";
  std::cout << "Temperature: " << temperature << "C\n";
  std::cout << "Dew Point: " << dew_point << "C\n";

  close(2);
  return 0;
}
```

This will require linking against WiringPi at compilation:

```
g++ -o path/to/output /path/to/script.cpp -l wiringPi
```

Ensure all dependencies are present on the Raspberry Pi, and the environmental monitoring system is thus complete.

## 4 Acknowledgments

This work has been funded by Canada's Institute of Particle Physics and Natural Sciences and Engineering Research Council.

# References


[1] Innovative Sensor Technology, *Humidity module textile filter*, https://www.ist-ag.com/en/products/humidity-module-textile-filter.

[2] Raspberry Pi Foundation, *The Raspberry Pi's GPIO pins*, https://projects.raspberrypi.org/en/projects/rpi-gpio-pins

[3] Raspberry Pi, *Getting started*, https://www.raspberrypi.com/documentation/computers/getting-started.html

[4] Kestrel Instruments, *What Is the Dew Point and How Is It Calculated?*, https://kestrelinstruments.com/blog/what-is-the-dew-point-and-how-is-it-calculated?

[5] GitHub, *WiringPi*, https://github.com/wiringpi/wiringpi