# DHT11 Temperature & Humidity Sensor Setup on Raspberry Pi 5 (System-Wide Installation)

This guide provides step-by-step instructions to connect and read data from a DHT11 temperature and humidity sensor using a Raspberry Pi 5. This method directly installs all necessary Python packages into your system's global Python environment, bypassing virtual environments.

Warning: This approach uses the --break-system-packages flag for pip installations. While functional for this specific project, it is generally not recommended by Raspberry Pi OS (due to PEP 668). This can potentially lead to conflicts with system-managed packages and may destabilize your operating system's Python installation. Proceed with caution and understand the implications.

# 1. Hardware Requirements

Before you begin, ensure you have the following:

Raspberry Pi 5 (8GB Recommended): Running the latest Raspberry Pi OS (Bookworm or newer).

DHT11 Temperature and Humidity Sensor:

3-pin module: (VCC, GND, DATA). These usually have a built-in pull-up resistor.

4-pin sensor: (VCC, DATA, NC, GND). You will require an external 10k Ohm pull-up resistor between the DATA pin and VCC.

Jumper Wires: Female-to-Female for connecting the sensor to the Pi's GPIO pins.

(Optional) Breadboard: For easier wiring, especially if using a 4-pin DHT11 and an external resistor.

Stable 5V Power Supply for your Raspberry Pi 5 (e.g., Raspberry Pi Official 27W USB-C PD PSU).

# 2. Hardware Connections (Wiring Diagram)

Connect your DHT11 sensor to the Raspberry Pi 5's GPIO header as follows. We will be using GPIO 17 for the data pin.

DHT11 VCC (or +)

rightarrow Raspberry Pi 3.3V Pin (Physical Pin 1 or 17)

DHT11 GND (or -)

rightarrow Raspberry Pi GND Pin (Physical Pin 6, 9, 14, 20, 25, 30, 34, or 39)

DHT11 DATA (or Out/S)

rightarrow Raspberry Pi GPIO 17 Pin (Physical Pin 11)

Pull-up Resistor (Important for 4-pin DHT11):

If you are using a bare 4-pin DHT11 sensor (without a small PCB/module), you must place a 10k Ohm resistor between the DHT11's DATA pin and its VCC pin. Most 3-pin DHT11 modules have this resistor integrated onto the small PCB.

(Please refer to online resources for a visual Raspberry Pi 5 GPIO pinout diagram if needed, ensuring you connect to 3.3V, GND, and Physical Pin 11 (BCM GPIO 17).)

# 3. Software Setup (System-Wide Installation)

This phase involves installing all necessary Python libraries and system dependencies directly onto your Raspberry Pi OS.

Open a Terminal:

Access your Raspberry Pi's terminal. This can be done directly on the desktop, via SSH, or VNC.

Update System Packages and Install Core Tools:

First, ensure your operating system's package list is up-to-date and install pip (Python's package installer) and libgpiod2 (a crucial GPIO access library).

Bash

sudo apt update

sudo apt upgrade -y

sudo apt install -y python3-pip libgpiod2

Install Adafruit Blinka's Dependencies System-Wide:

Blinka is a compatibility layer that allows CircuitPython libraries (like adafruit\_dht) to run on Linux-based single-board computers. Its installer script has its own Python dependencies. We use --break-system-packages to allow these to be installed globally.

Bash

sudo pip3 install --break-system-packages --upgrade setuptools adafruit-python-shell

Download and Run the Adafruit Blinka Installer Script:

This script performs essential system-level configurations to enable GPIO communication.

Bash

wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/raspi-blinka.py

sudo python3 raspi-blinka.py

Follow any on-screen prompts. The script may ask you to confirm enabling certain interfaces (e.g., I2C, SPI). Respond with y and press Enter when prompted.

Crucially, if the script prompts you to reboot your Raspberry Pi, do so immediately:

Bash

sudo reboot

After Rebooting, Install DHT and Requests Libraries System-Wide:

Once your Raspberry Pi has restarted, open a new terminal session. Now, install the specific Python libraries for your DHT sensor and optional ThingSpeak integration.

Bash

sudo pip3 install --break-system-packages adafruit-circuitpython-dht requests

adafruit-circuitpython-dht: The core library for interacting with DHT sensors.

requests: A library for making HTTP requests, used here for sending data to ThingSpeak.

# 4. Python Script for DHT11 Readings

This script will read temperature and humidity from the DHT11 sensor every 5 seconds and print it to the console. It also includes optional functionality to upload data to ThingSpeak.

Create the Python Script File:

Open a text editor on your Raspberry Pi (e.g., Thonny, Nano, VS Code).

Create a new file and save it as dht11\_reader.py in your desired directory (e.g., /home/pi/).

Paste the following code into dht11\_reader.py:

Python

import time

import board

import adafruit\_dht

import requests # Used for ThingSpeak upload

# --- ThingSpeak Configuration (Optional - only if uploading to cloud) ---

# IMPORTANT: Replace "YOUR\_THINGSPEAK\_WRITE\_API\_KEY" with your actual API key

THINGSPEAK\_API\_KEY = "YOUR\_THINGSPEAK\_WRITE\_API\_KEY"

THINGSPEAK\_URL = "https://api.thingspeak.com/update"

# ---------------------------------------------------------------------

# Initialize DHT11 on GPIO17 (BCM pin 17, Physical Pin 11 on the Pi header)

# Ensure your DHT11 Data pin is physically connected to this GPIO.

try:

dht\_device = adafruit\_dht.DHT11(board.D17)

print("DHT11 sensor initialized on GPIO17.")

except ValueError as e:

print(f"ERROR: Problem initializing DHT sensor on GPIO17: {e}")

print("Please check your wiring, selected GPIO pin (BCM 17), and ensure the sensor type is DHT11.")

print("Exiting script as sensor could not be detected/initialized.")

exit() # Exit the script immediately if initialization fails

print("Starting DHT11 sensor readings...")

while True:

try:

temperature\_c = dht\_device.temperature

humidity = dht\_device.humidity

if temperature\_c is not None and humidity is not None:

temperature\_f = temperature\_c \* (9 / 5) + 32

# Print to console with proper degree symbol

print(f"Read OK: Temp: {temperature\_c:.1f}\u00b0C | {temperature\_f:.1f}\u00b0F | Humidity: {humidity:.1f}%")

# --- Send data to ThingSpeak (Optional) ---

# This block only executes if you've provided a valid ThingSpeak API Key

if THINGSPEAK\_API\_KEY != "YOUR\_THINGSPEAK\_WRITE\_API\_KEY":

payload = {

'api\_key': THINGSPEAK\_API\_KEY,

'field1': temperature\_c,

'field2': humidity

}

try:

response = requests.post(THINGSPEAK\_URL, data=payload)

if response.status\_code == 200:

print("ThingSpeak: Data sent successfully!")

else:

print(f"ThingSpeak: Error sending data. Status: {response.status\_code}, Response: {response.text}")

except requests.exceptions.RequestException as e:

print(f"ThingSpeak: Network error: {e}")

# -------------------------------------------

else:

# This case happens if the sensor returns None, but doesn't throw a RuntimeError

print("Read Attempt: Failed to retrieve valid data (returned None). Retrying...")

except RuntimeError as error:

# Catches common DHT read errors (e.g., sensor not ready, connection issues)

print(f"CRITICAL DHT Sensor Read Error: {error.args[0]}")

print("Sensor is consistently unresponsive or connection is bad. Exiting script.")

try:

dht\_device.exit() # Clean up sensor resources

except AttributeError:

pass # Ignore if dht\_device doesn't have exit method or not fully initialized

exit() # Exit the script due to persistent error

except Exception as error:

# Catches any other unexpected critical errors

print(f"An unexpected critical error occurred: {error}")

print("Exiting script due to unexpected error.")

try:

dht\_device.exit()

except AttributeError:

pass

exit() # Exit the script

time.sleep(5) # Wait 5 seconds before the next reading (DHT11 minimum interval is > 1s)

# 5. Running the Script

Open a Terminal: Navigate to the directory where you saved dht11\_reader.py.

Execute the Script:

Bash

python3 dht11\_reader.py

Monitor the Output:

If successful, you will see temperature and humidity readings printed every 5 seconds.

If the sensor cannot be initialized or consistently fails to read, the script will print an error message and exit, indicating a hardware or wiring problem.

# 6. (Optional) ThingSpeak Integration

To visualize your sensor data online:

Create a ThingSpeak Account and Channel:

Go to ThingSpeak.com and create a free account.

Log in and create a new channel: Channels

rightarrow My Channels

rightarrow New Channel.

Provide a Name (e.g., "Raspberry Pi Weather").

Name Field 1 as Temperature (C) and Field 2 as Humidity (%).

Click "Save Channel."

Retrieve Your Write API Key:

Navigate to the "API Keys" tab for your newly created channel.

Copy the "Write API Key".

Update Your Python Script:

Edit your dht11\_reader.py file.

Find the line: THINGSPEAK\_API\_KEY = "YOUR\_THINGSPEAK\_WRITE\_API\_KEY"

Replace "YOUR\_THINGSPEAK\_WRITE\_API\_KEY" with the actual Write API Key you copied from ThingSpeak.

Save the changes to the script.

Run the Script Again:

Execute the script as described in Section 5.

Check your ThingSpeak channel's "Private View" (or "Public View" if enabled). You should see your temperature and humidity data appearing on the charts as the script sends readings.

Note on ThingSpeak Free Tier: ThingSpeak's free tier typically has a rate limit of one update every 15 seconds. While your script reads every 5 seconds, ThingSpeak will only process updates at its allowed frequency, discarding more frequent submissions. If you require higher update rates, consider a paid ThingSpeak plan or an alternative IoT platform.