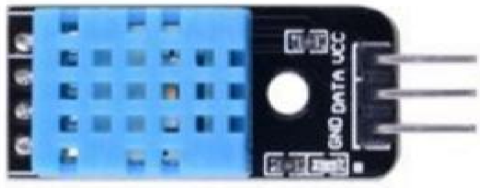
# Temperature and Humidity Sensor

# (aka DHT11)



## Overview

The DHT11 module combines a capacitive humidity sensor and an NTC thermistor to measure ambient air humidity and temperature, with high reliability and excellent long-term stability. An onboard microcontroller allows it to outputs to a single digital pin, so no analog-to-digital conversion required. This experiment shows how to query the DHT11's sensor readings and display them through the Raspberry Pi's command line interface.

## Experimental Materials

| | |
|---|---|
| Raspberry Pi | x1 |
| Breadboard | x1 |
| DHT11 module | x1 |
| Dupont jumper wires | |

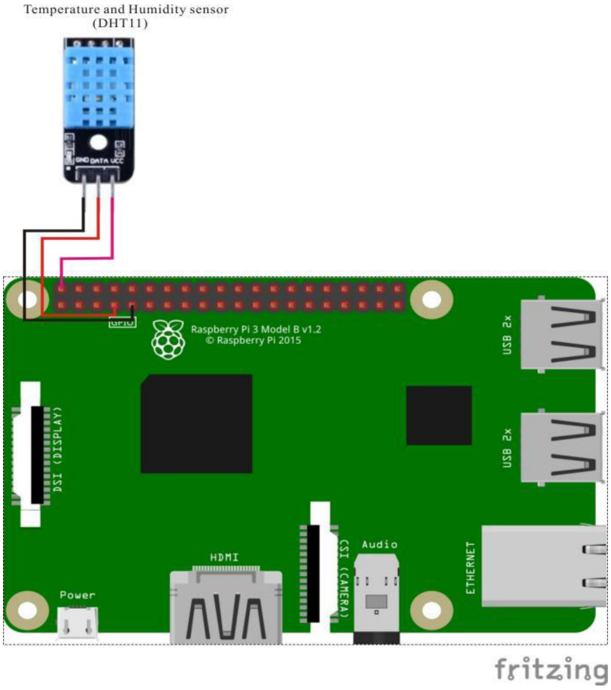
## Experimental Procedure

1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.
2. Install the DHT11 temperature and humidity sensor on your breadboard, and use Dupont jumper wires to connect it to your Raspberry Pi as illustrated in the Wiring Diagram below.
3. Execute the sample code stored in this experiment's subfolder.
   If using C, compile and execute the C code:
   ```
   cd Code/C
   gcc dht11.c -o dht11.out -lwiringPi
   ./dht11.out
   ```

   If using Python, launch the Python script:
   ```
   cd Code/Python
   python dht11.py
   ```

4. Make experimental observations. The temperature and relative humidity readings are displayed on the Raspberry Pi command line interface and repeatedly updated. For more details on communicating with the DHT11 from software, search the internet for “DHT11 datasheet.”

## Wiring Diagram



Temperature & Humidity Sensor (DHT11) pin position:

| | | |
|---|---|---|
| DATA | ↔ | Raspberry Pi pin 7 |
| VCC | ↔ | Raspberry Pi +5V |
| GND | ↔ | Raspberry Pi GND |

## Sample Code

### Python Code

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
def collect():
    THdata = []
    channel = 7
    data = []
    GPIO.setmode(GPIO.BOARD)
    time.sleep(2)
    GPIO.setup(channel, GPIO.OUT)
    GPIO.output(channel, GPIO.LOW)
    time.sleep(0.02)
    GPIO.output(channel, GPIO.HIGH)
    GPIO.setup(channel, GPIO.IN)
    while GPIO.input(channel) == GPIO.LOW:
        continue
    while GPIO.input(channel) == GPIO.HIGH:
        continue
    j = 0
    while j < 40:
        k = 0
        while GPIO.input(channel) == GPIO.LOW:
            continue
        while GPIO.input(channel) == GPIO.HIGH:
            k += 1
            if k > 100:
                break
        if k < 8:
            data.append(0)
        else:
            data.append(1)
        j += 1

    # print("sensor is working.")
    # print(data)
    humidity_bit = data[0:8]
    humidity_point_bit = data[8:16]
    temperature_bit = data[16:24]
    temperature_point_bit = data[24:32]
    check_bit = data[32:40]
    humidity = 0
    humidity_point = 0
    temperature = 0
    temperature_point = 0
    check = 0
    for i in range(8):
        humidity += humidity_bit[i] * 2 ** (7 - i)
        humidity_point += humidity_point_bit[i] * 2 ** (7 - i)
        temperature += temperature_bit[i] * 2 ** (7 - i)
        temperature_point += temperature_point_bit[i] * 2 **
(7 - i)
        check += check_bit[i] * 2 ** (7 - i)
    tmp = humidity + humidity_point + temperature + 
temperature_point
    if check == tmp:
        print 
"temperature:%d.%d" %(temperature,temperature_point),"C","
humidity :", humidity, "%"
        THdata.append(temperature)
        THdata.append(humidity)
        return THdata
    else:
        # print("wrong")
        time.sleep(1)
        return collect()

while True:
    rHdata = collect()
    time.sleep(3)
```

## C Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>

typedef unsigned char uint8;
typedef unsigned int  uint16;
typedef unsigned long uint32;

#define HIGH_TIME 32

int pinNumber = 7;
uint32 databuf;

uint8 readSensorData(void)
{
    uint8 crc;
    uint8 i;

    pinMode(pinNumber, OUTPUT); // set mode to output
    digitalWrite(pinNumber, 0); // output a high level
    delay(25);
    digitalWrite(pinNumber, 1); // output a low level
```

```
   pinMode(pinNumber, INPUT); // set mode to input
   pullUpDnControl(pinNumber, PUD_UP);
   delayMicroseconds(27);
   if(digitalRead(pinNumber) == 0) //SENSOR ANS
   {
       while(!digitalRead(pinNumber)); //wait to high
      for(i=0;i<32;i++)
      {
        while(digitalRead(pinNumber)); //data clock start
        while(!digitalRead(pinNumber)); //data start
        delayMicroseconds(HIGH_TIME);
        databuf*=2;
         if(digitalRead(pinNumber)==1) //1
         {
           databuf++;
         }
       }
      for(i=0;i<8;i++)
       {
        while(digitalRead(pinNumber)); //data clock start
        while(!digitalRead(pinNumber)); //data start
        delayMicroseconds(HIGH_TIME);
        crc*=2;
        if(digitalRead(pinNumber)==1) //1
         {
             crc++;
         }
       }
      return 1;
   }
   else
   {
      return 0;
   }
}
int main (void)
{
  if (-1 == wiringPiSetup())
  {
   printf("Setup wiringPi failed!");
   return -1;
  }
  pinMode(pinNumber, OUTPUT); // set mode to output
  digitalWrite(pinNumber, 1); // output a high level
   while(1)
   {
    pinMode(pinNumber,OUTPUT); // set mode to output
    digitalWrite(pinNumber, 1); // output a high level
    delay(3000);
    if(readSensorData())
    {
       printf("Congratulations ! Sensor data read ok!\n");

printf("RH:%d.%d%\n",(databuf>>24)&0xff,(databuf>>16)&0xf
f);

printf("TMP:%d.%dC\n",(databuf>>8)&0xff,databuf&0xff);
       databuf = 0;
   }
   else
   {
      printf("Sorry! Sensor does not respond!\n");
      databuf = 0;
   }
  }
  return 0;
}
```

## Technical Background

- ◆ Working voltage: 5VDC/3.3VDC.
- ◆ Humidity measurement range: 20%~90%RH.
- ◆ Measurement accuracy: ±5%RH.
- ◆ Temperature measurement range: 0 ~ 50 ˚C
- ◆ Temperature measuring accuracy: ±2 ˚C
- ◆ Data port equipped with a pull resistor
- ◆ 3mm fixed screw hole for easy installation