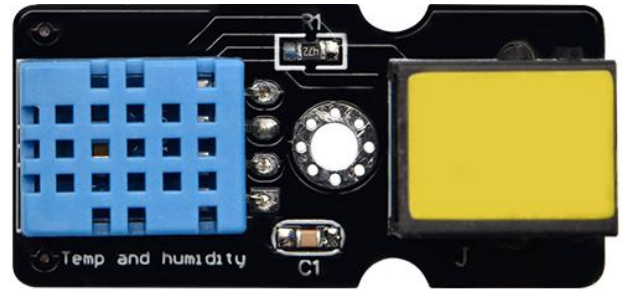


# DHT11 Temperature and Humidity Sensor (000x0000 Article Number) (TS2153)



## Product Details

The TelePort DHT11 is a digital temperature and humidity composite sensor with calibrated digital signal outputs. It contains a resistive humidity sensing element and NTC temperature measuring component with the advantages of low power consumption and long-term stability. Additionally, it can output single bus digital signals through built-in ADC, immensely save the I/O ports of the control board.



## Features and Benefits

- Compatible with RJ11 6P6C OKdo TelePort Control boards and expansion shields.
- The DHT11 sensor is a combined temperature and humidity device with pre-calibrated digital output.
- Provides fast response, low-interference and long term stability.

## Technical Specifications

Sensor type	Digital input
Working voltage	5V
Relative Humidity and temperature measurement	Good for 20-90% humidity readings with 5% accuracy; Good for 0-50°C temperature readings $\pm 2^{\circ}\text{C}$ accuracy
Dimensions	42mm*20mm*18mm
Weight	5.6g

## Applications

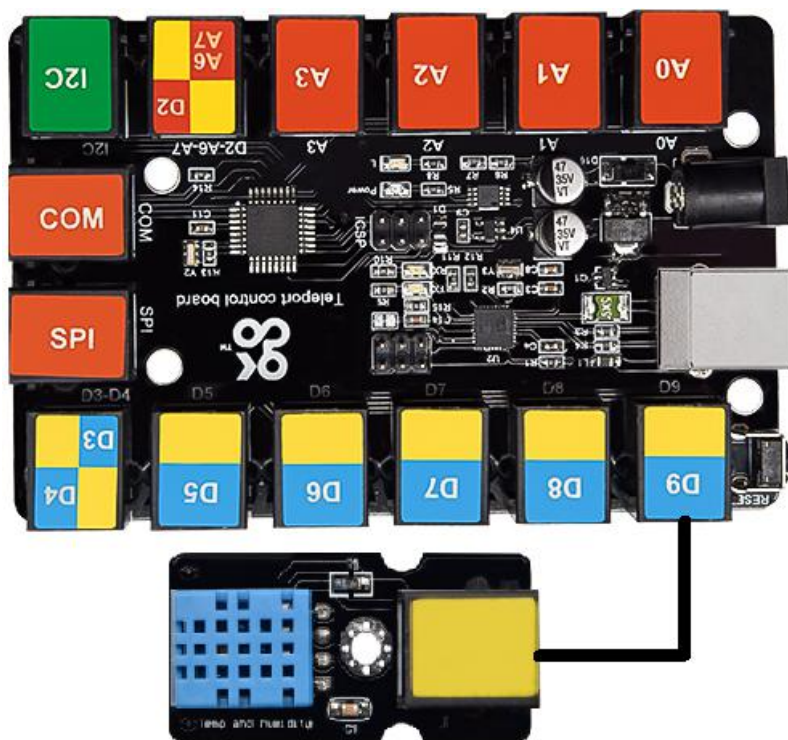
- Weather station
- Automatic control
- Testing and inspection equipment
- Soil moisture detector

## Comparison

Project	DHT11	DHT22
Humidity		
Range	5 ~ 95%RH	0 ~ 100%RH
Accuracy	±5% (Typical)	±2% (Typical)
Temperature		
Range	-20 ~ 60°C	-40 ~ 80°C
Accuracy	±2% (Typical)	±0.5% (Typical)

This module is compatible with the TS2180-Raspberry Pi shield, the TS2179-Micro:bit shield and the TS2178-TelePort main board.

### ➤ Arduino Application



This module is compatible with the TS2178 TelePort control board.

### Test Code

Before compiling test code, remember to place libraries you need in the libraries of Arduino IDE.

Unzip the library files, that is, copy the unzipped **Dht11** folder into the libraries of Arduino IDE.

After pasting it, reboot the compiler.

For instance: C:\Program Files\Arduino\libraries

```
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 9
void setup(){
  Serial.begin(9600);
  Serial.println("DHT TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
}
void loop(){
  int chk;
  Serial.print("DHT11, \t");
  chk = DHT.read(DHT11_PIN);//READ DATA
  switch (chk){
    case DHTLIB_OK:
      Serial.print("OK,\t");
      break;
    case DHTLIB_ERROR_CHECKSUM:
      Serial.print("Checksum error,\t");
      break;
    case DHTLIB_ERROR_TIMEOUT:
      Serial.print("Time out error,\t");
      break;
    default:
      Serial.print("Unknown error,\t");
      break;
  }
  // DISPLAT DATA
  Serial.print(DHT.humidity,1);
  Serial.print(",\t");
  Serial.println(DHT.temperature,1);
  delay(1000);
}
```

### Test Result

Wire up, upload code, power it up, open serial monitor and set baud rate to 9600. Then the serial monitor will show the temperature and humidity value.

```
COM9
Send

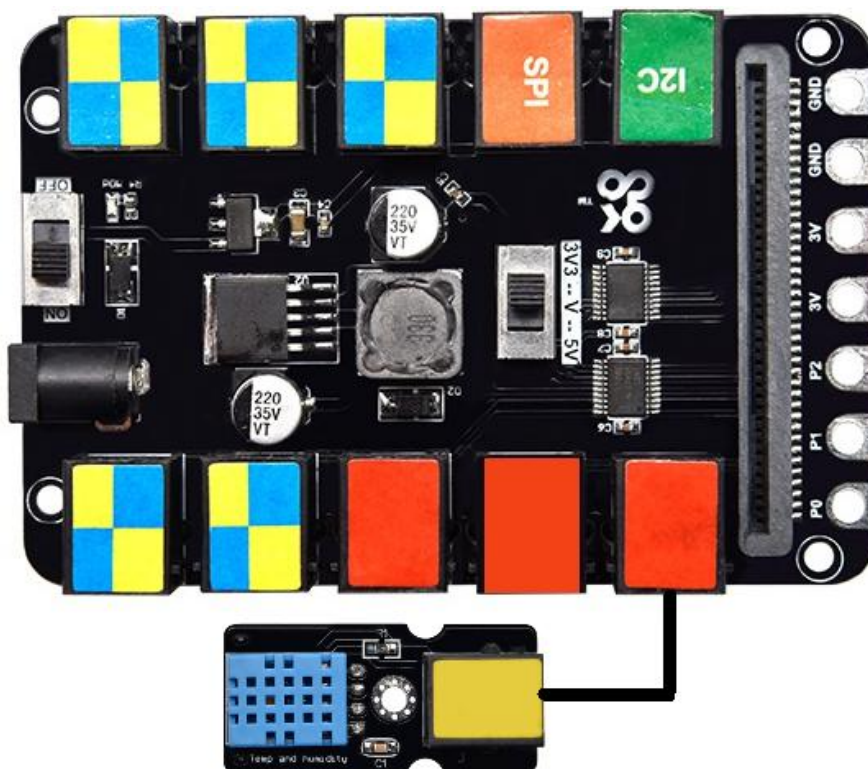
DHT TEST PROGRAM
LIBRARY VERSION: 0.4.1

Type,    status, Humidity (%),    Temperature (C)
DHT11,   Checksum error, 80,      33
DHT11,   Time out error, 80,      33
DHT11,   Checksum error, 81,      33
DHT11,   Time out error, 81,      33
DHT11,   Checksum error, 82,      33
DHT11,   Time out error, 82,      33
DHT11,   Checksum error, 82,      33
DHT11,   Time out error, 82,      33
DHT11,   Checksum error, 76,      33
DHT11,   Time out error, 76,      33
```

☒ Autoscroll ☐ Show timestamp    Newline    9600 baud    Clear output

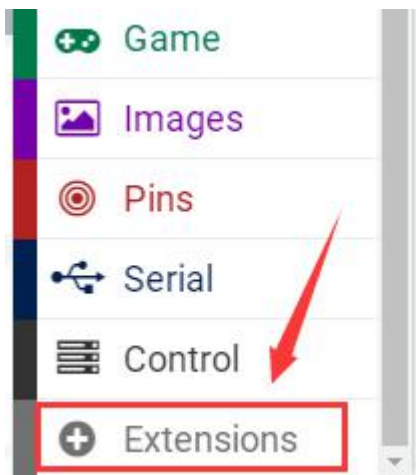
If you want to know more details about Arduino and the TelePort control board, you can refer to TS2178.

### ➤ Micro:bit Application

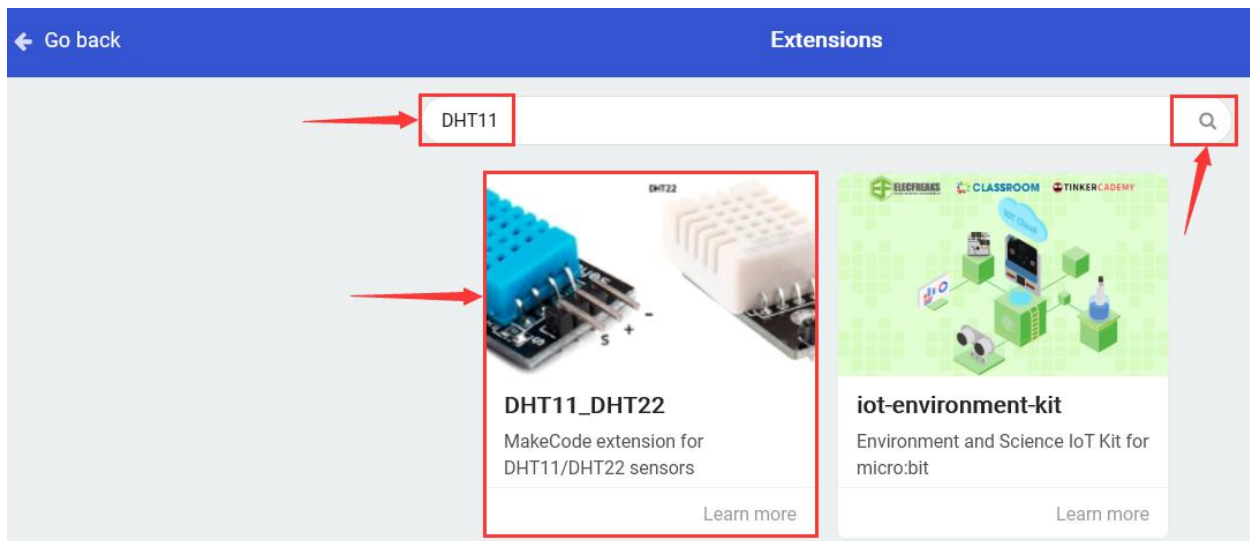


It is compatible with the Micro:bit board and the TS2179 Micro:bit expansion board.

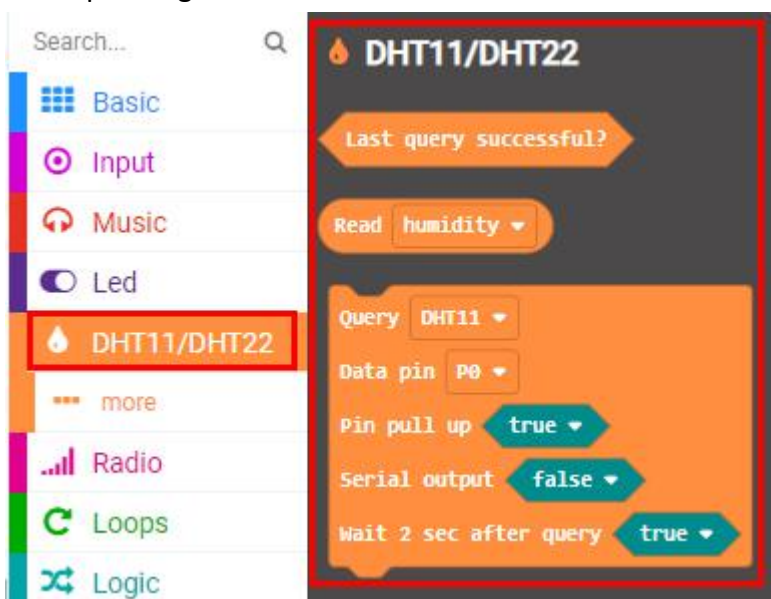
Add the library of the DHT11 temperature and humidity sensor, as shown below;  
Use the library file to set code, click“Extensions”



Enter **DHT11** to search, as shown below, click the library file and download it automatically.



After the library of the DHT11 temperature and humidity sensor is installed, then you can view the corresponding block in the blocks list.



## Test Code



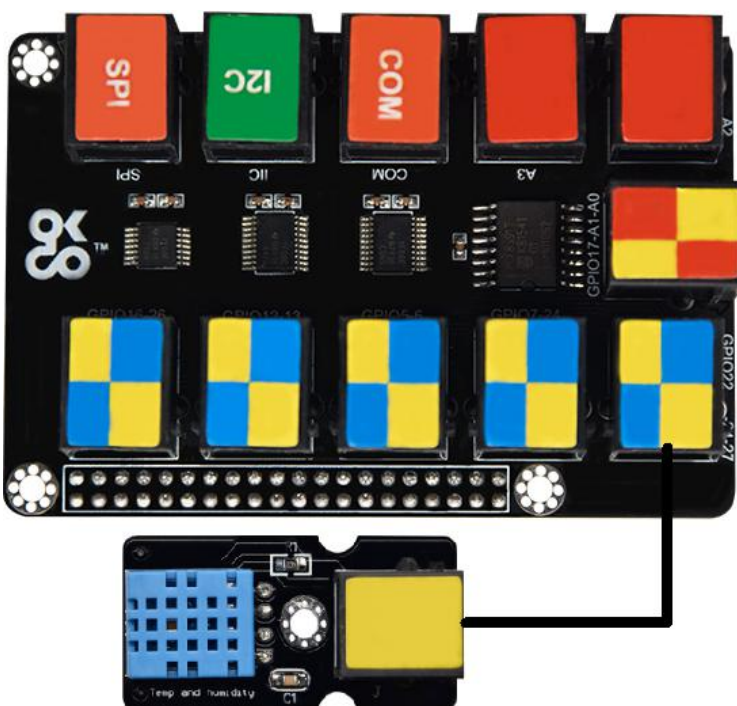
- .....①Run the “on start” block to boot the program
- .....②Open the LED matrix of the Micro:bit
- .....③The program is run circularly under the command of “forever” block
- .....④Check the data from P0 of the DHT11 sensor. If you use a 4-pin/no pcb version, you need to unplug the data pin. It is recommended to wait 1 (DHT11) or (DHT22) 2 seconds for each check
- .....⑤the Micro:bit shows the displayed temperature
- .....⑥the Micro:bit shows the displayed humidity

## Test Result

Wire up, insert the Micro:bit V2.0 into the shield, turn DIP switch to 3V3, upload test code and power it up. The Micro:bit will show the ambient temperature and humidity.

If you want to know more details about the Micro:bit board and Micro:bit shield, you can refer to TS2179.

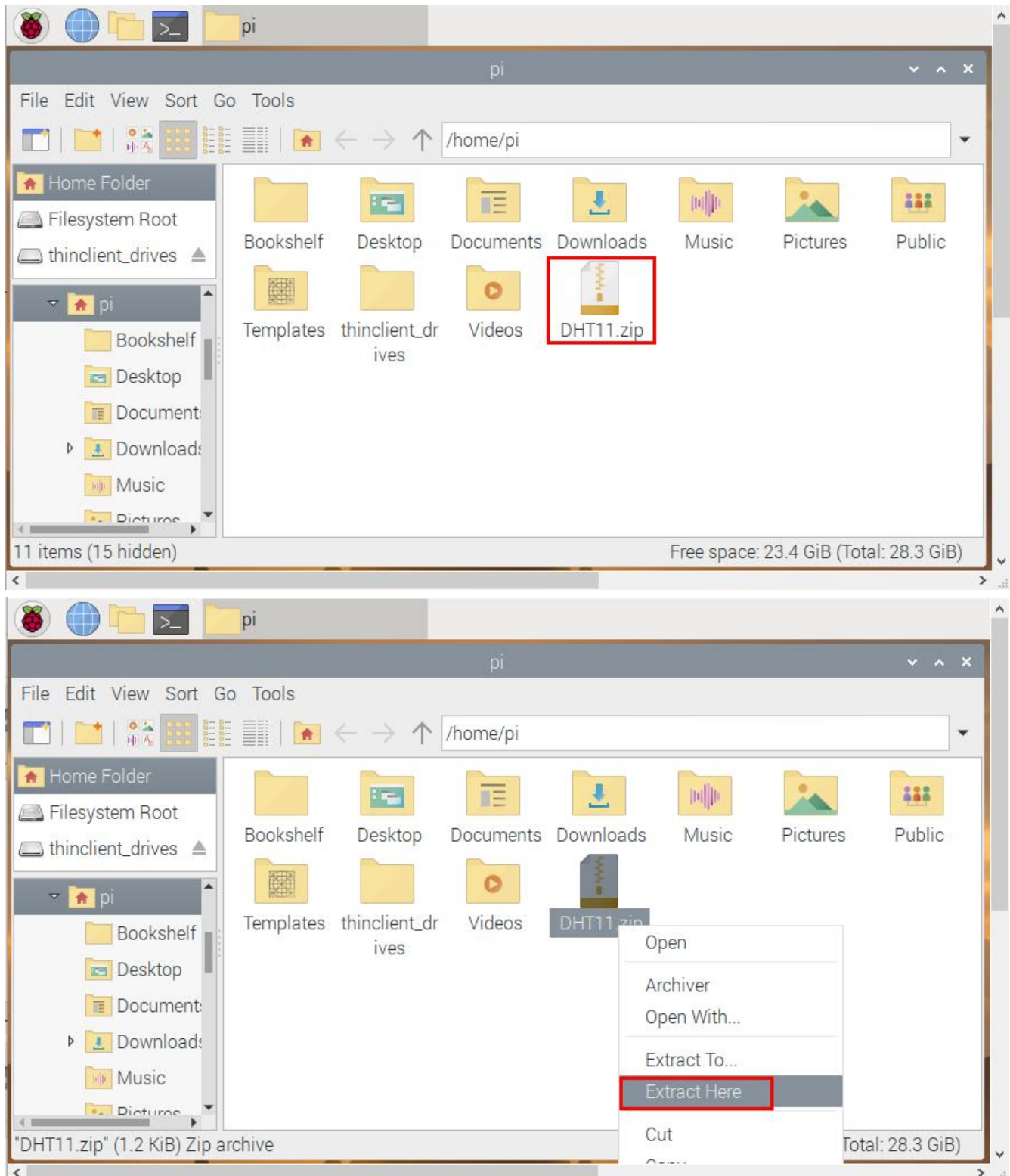
### ➤ Raspberry Pi Application

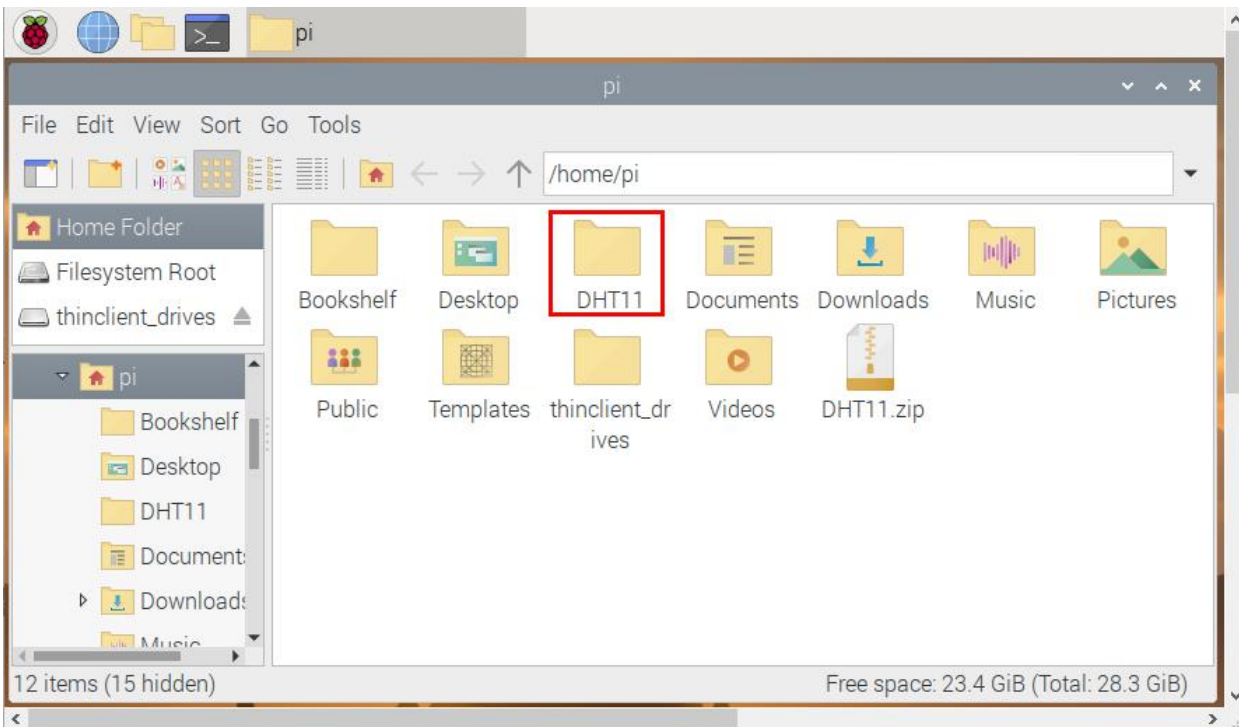


This module is compatible with the Raspberry Pi board and the TS2180 Raspberry Pi shield.

### Copy the test code to Raspberry Pi system to run it

(1) Save the test code in the **pi** folder of Raspberry Pi system. Then place the **DHT11.zip** file we provide in the **pi** folder, right-click and click **Extract Here**. As shown below:





(2) Compile and run test code:

Input the following code and press "Enter"

```
cd /home/pi/DHT11
gcc DHT11.c -o DHT11 -lwiringPi
sudo ./DHT11
```

(3) Test Result:

Insert the shield into the Raspberry Pi board. After programming finishes, then the terminal will display the detected temperature and humidity value.

Note: press Ctrl + C to exit code running

```
File Edit Tabs Help
pi@raspberrypi:~ $ cd /home/pi/C_code/DHT11
pi@raspberrypi:~/C_code/DHT11 $
pi@raspberrypi:~/C_code/DHT11 $ gcc DHT11.c -o DHT11 -lwiringPi
pi@raspberrypi:~/C_code/DHT11 $
pi@raspberrypi:~/C_code/DHT11 $ sudo ./DHT11
RH:50,TEMP:29
RH:49,TEMP:29
RH:54,TEMP:29
RH:52,TEMP:29
RH:49,TEMP:29
RH:48,TEMP:29
RH:50,TEMP:30
RH:56,TEMP:30
RH:95,TEMP:30
```

## Test Code

File name: **DHT11.c**

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

#define MAX_TIME 85
#define DHT11PIN 3 //BCM GPIO 22
#define ATTEMPTS 5 //retry 5 times when no response
int dht11_val[5]={0,0,0,0,0};

int dht11_read_val(){
    uint8_t lststate=HIGH; //last state
    uint8_t counter=0;
    uint8_t j=0,i;
    for(i=0;i<5;i++)
        dht11_val[i]=0;

    //host send start signal
    pinMode(DHT11PIN,OUTPUT); //set pin to output
    digitalWrite(DHT11PIN,LOW); //set to low at least 18ms
    delay(18);
    digitalWrite(DHT11PIN,HIGH); //set to high 20-40us
    delayMicroseconds(40);

    //start receive dht response
    pinMode(DHT11PIN,INPUT); //set pin to input
    for(i=0;i<MAX_TIME;i++)
    {
        counter=0;
        while(digitalRead(DHT11PIN)!=lststate){ //read pin state to see if dht responded. if dht always high for 255 +
            counter++;
            delayMicroseconds(1);
            if(counter==255)
                break;
        }
        lststate=digitalRead(DHT11PIN); //read current state and store as last state.
        if(counter==255) //if dht always high for 255 + 1 times, break this for circle
            break;
        // top 3 transistions are ignored, maybe aim to wait for dht finish response signal
        if((i>=4)&&(i%2==0)){
            dht11_val[j/8]<=1; //write 1 bit to 0 by moving left (auto add 0)
            if(counter>16) //long mean 1
                dht11_val[j/8]|=1; //write 1 bit to 1
```

```

        j++;
    }
}
// verify checksum and print the verified data
if((j>=40)&&(dht11_val[4]==((dht11_val[0]+dht11_val[1]+dht11_val[2]+dht11_val[3])& 0xFF))){
    printf("RH:%d,TEMP:%d\n",dht11_val[0],dht11_val[2]);
    return 1;
}
else
    return 0;
}

int main(void){
    int attempts=ATTEMPTS;
    if(wiringPiSetup()==-1)
        exit(1);
    while(1)
    {
        dht11_read_val(); //get result including printing out
        delay(3000); //The read cycle needs to be greater than 2 seconds
    }
    return 0;
}

```

If you want to know how to utilize Raspberry Pi and the Raspberry Pi shield, you can refer to TS2180.

\*\*\*END\*\*\*