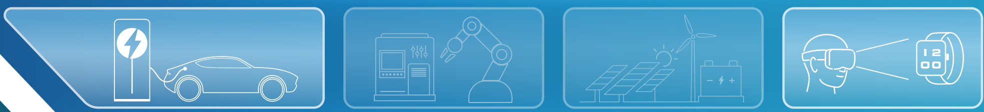
Application Note

NOVOSENSE

Relative Humidity and Temperature Sensor

AN-12-0037

Author： Juanjuan Shao



AN-12-0037

NOVOSENSE

Relative Humidity and Temperature Sensor

Product Overview

NSHT30 is a relative humidity and temperature sensor with I²C Interface. It is a CMOS-MEMS sensor chip based on Novosense's new humidity and temperature platform. NSHT30 integrated a capacitive-based relative humidity sensor, a CMOS temperature sensor, a signal processor and a high speed I²C interface in a single chip and packaged in a small package. The outline of the LGA/DFN package is only 2.5 mm in length, 2.5 mm in width and 0.9 mm in height. This allows the NSHT30 to be more widely integrated into a variety of applications.

The application note provides operational guidance, software configuration, and hardware design guidelines. It can help users familiarize the NSHT30.

Applications

- Washer & dryer
- Smart thermostats and room monitors
- White goods
- Printers
- Humidifier/dehumidifier
- Air quality monitoring
- Wireless sensor
- Home appliances

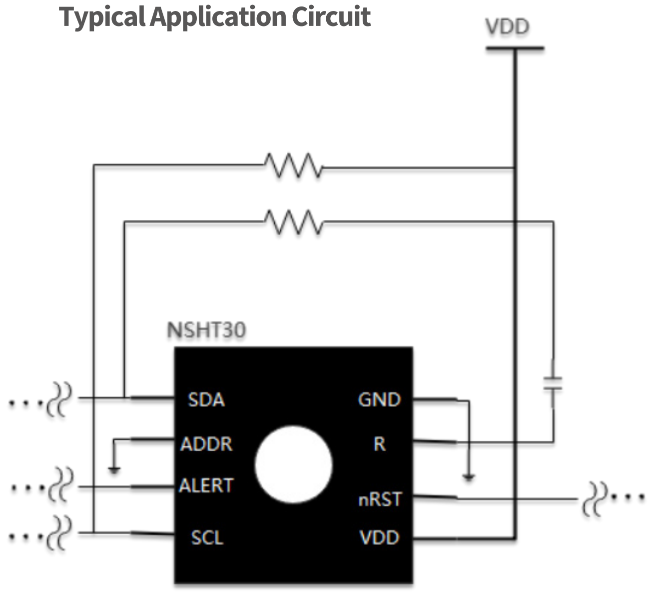
Device Information

- NSHT30-CLAR 2.5mm×2.5mm×0.9mm
- NSHT30-QDNR 2.5mm×2.5mm×0.9mm

Key Features

- Relative Humidity Sensor:
  - -Humidity Range: 0%RH~100% RH
  - -Humidity Accuracy: ±3% RH (Typ)
- Temperature Sensor:
  - -Temperature Range: -40℃~125℃
  - -Temperature Accuracy: ±0.3℃ (Typ)
- Relative humidity and temperature compensated digital output
- Wide supply voltage range, from 2 V to 5.5 V
- I2C Interface with communication speeds up to 1 MHz
  - -Two selectable addresses
  - -Data protection with CRC checksum
- Average Current: 3.2µA
- Two selectable LGA/DFN package

Typical Application Circuit



INDEX

1.NSHT30 Devices

1.1.LGA Package NSHT30-CLAR

NSHT30 integrated a capacitive-based relative humidity sensor, a CMOS temperature sensor, a signal processor and a high speed I2C interface in a single chip and packaged in a small LGA package. The outline of the LGA package is only 2.5 mm in length, 2.5 mm in width and 0.9 mm in height.

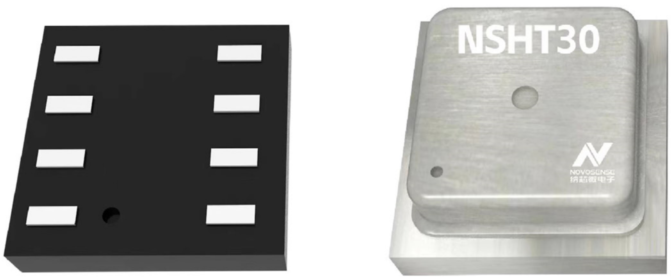


Figure 1.1 LGA Package

1.2.DFN Package NSHT30-QDNR

The NSHT30 have two different package, the other is DFN Open Cavity package. The outline of the DFN package is only 2.5 mm in length, 2.5 mm in width and 0.9 mm in height.

Figure 1.2 DFN Package

2.Operation and Function Description

2.1.Operation and Function Description

2.3.1.Pressure data processing

The NSHT30 have two different measurement mode triggered by different measurement commands. There are a variety of measurement commands can be selected in single shot mode. As shown as Table 2.1, it gives different 16-bit measurement commands for single shot mode. The main differences between commands are repeatability (low, medium, high).

Table 2.1 Measurement commands in single shot mode

Hex. code		**Condition**
MSB	LSB	Repeatability
24	00	High
	0B	Medium
	16	Low

In single shot mode, after the sensor received the complete measurement sequence, it will start the measurement of temperature and relative humidity in the environment. Once the measurement is complete, the data of the measurement is updated in the sensor. Then, users can readout the data from the sensor. The data is made up of 6 bytes: 2 bytes of temperature data, 1 byte of temperature CRC checksum, 2 bytes of humidity data, and 1 byte of humidity CRC checksum (in this order).

For more descriptions, please refer to the document of Datasheet or Demo Code.

2.3.1.Pressure data processing

Periodic data acquisition mode is quite different with single shot mode. The sensor will conduct a periodic measurement and update the measurement result itself in this mode. As a mode of periodic data acquisition, 5 different periodic can be used, and there are 3 different repeatability choices for each periodic. Then, a total of 15 different measurement commands can be selected in periodic data acquisition mode. As shown in Table 2.2, the corresponding 16-bit commands are listed. The periodic of each mode is described with mps (measurement per second).

Table 2.2 Measurement commands for periodic data acquisition mode

Hex. code		**Condition**	
MSB	LSB	Periodic	Repeatability
20	32	0.5 mps	High
	24		Medium
	2F		Low
21	30	1 mps	High
	26		Medium
	2D		Low
22	36	2 mps	High
	20		Medium
	2B		Low
23	34	4 mps	High
	22		Medium
	29		Low
27	37	10 mps	High
	21		Medium
	2A		Low

The master can acquire measurement data through the read data command (hex code:E0 00). The data read back from the sensor is the same as the Single Shot Data Acquisition Mode.

When the sensor was already in a periodic mode, users are recommended to send the break command (hex code:30 93) before sending the new measurement command.

2.2.Function Description

2.2.1.Heater

The function of NSHT30 internal heater is only used for plausibility check. The heater will cause a temperature rise and humidity down of the sensor after being enabled.

Table 2.3 Heater command

Hex. code		Command
MSB	LSB	
30	6D	Heater on
	66	Heater off

The Heater function can be applied to the following conditions:

1.The heater will cause a temperature rise and humidity down of the sensor after being enabled. The temperature rise of the heater is related to many parameters, usually is in the range of several degrees centigrade. Therefore, it can determine whether the sensor is in a normal working state

2.When the measurement environment is too low, it can cause condensation on the sensor surface. The heater on can help the sensor return to normal state.

2.2.2.Reset

The NSHT30 system reset can be generated by delivering a command (soft reset) or by sending a pulse to the special reset pin (nReset pin). Moreover, a system reset is also generated internally during power-up. The sensor will not process commands during the reset procedure. The reset caused by the nReset Pin is a full reset of the sensor like repower the sensor. It’s a better choice than removing the power supply to get a full reset of the sensor.

Table 2.4 soft reset command

Hex. code		Command
MSB	LSB	
30	A2	Soft reset

Besides, it is not recommended that customers reset the sensor on the I2C bus through the General Call. NSHT30 will respond to the General Call reset command, but there are special timing characteristics. If you require to use the General Call command, please contact our company for relevant support.

2.3.Notes for Attention

2.3.1.Recommended operating conditions

The recommended operating environment of NSHT30 should be controlled within the temperature range of 0-80 °C, and the relative humidity range of 20%RH - 80% RH, because its performance is the best at this range.

2.3.2.Avoid chemical substance

NSHT30 are susceptible to pollutants and must be protected from exposure to volatile chemicals, acids, alkalis, and cleaning agents. Ketones, acetone, ethanol, isopropanol, toluene may cause irreversible drift in sensor relative humidity accuracy. Thus, these guidelines must be followed:

1.Avoid exposing the sensor to volatile chemicals such as solvents or organic compounds. These chemicals are typically present in epoxy resins, adhesives, adhesives, and plastics. They all have potential impacts on sensors.

2.Avoid contact with strong acids and alkalis.

3.Avoid using cleaning agents such as PCB board cleaning agent, high concentration alcohol, etc.

2.3.3.Recovery method under extreme conditions

If the sensor is exposed outside the recommended operating conditions for a long time (over 60 hours), especially when the relative humidity is less than 3% RH or greater than 80% RH, the humidity sensors may temporarily get a negative humidity offset. The offset will slowly disappear by itself when the sensor is exposed to ambient conditions. To avoid the offset in the relative humidity readings, the sensor element must be rehydrated after the soldering process. The recommended rehydration conditions are either:

- Baking: A relative humidity< 5% RH at 95℃ for at least 12 hours
- Rehydration: A relative humidity of 90% RH at 25℃ for at least 12 hours
- At room temperature for at least 12 hours

2.3.4.Long term drift

NSHT30 long term drift can be calculated according to the following formula.

$$AF = AF_H * AF_T = (\frac{H_A}{H_U})^n * e^{\frac{E_a}{K}(\frac{1}{T_U} - \frac{1}{T_A})}$$

AF is the acceleration factor of the acceleration conditions relative to the environment; is acceleration factor of humidity; is acceleration factor of temperature. is model parameter, it is determined by the corrosion characteristics, and the recommended value range is 1-5; K is the Boltzmann constant; is the thermal activation energy(0.65 eV);
 is the relative humidity of the accelerated test; is the relative humidity of the operating environment; is the absolute temperature of the accelerated test; is the absolute temperature of the operating environment;
NSHT30 long term drift is based on the results of HTOL.

3.Storage and Operation

3.1.ESD

To ensure the normal operation of the sensor, it is necessary to prevent electrostatic discharge (ESD) at all times. All processing of sensors should be carried out specifically within the Electrostatic Discharge Protection Area (EPA) to ensure that the protection area has been correctly set up to minimize the risk of ESD. This includes grounding personnel with wrist straps or similar measures, and grounding all conductive materials.

In addition, all operations should be carried out on a grounded conductive floor. To further protect the sensor, ESD protective materials should be used for packaging when not being processed in EPA.

Figure 3.1 ESD Protection

3.2.Storage

NSHT30 is packaged and stored using sealed anti-static tape. After removing the tape, the sensor can be stored in a humidity and temperature controlled environment before use. The limits for storage temperature and humidity depend on the humidity sensitivity level (MSL) of the sensor, with NSHT30 CLAR following MSL3 and NSHT30 QDNR following MSL1.

Do not store the NSHT30 in anti-static polyethylene bags or packaging materials (pink foam/packaging), as these materials will emit gases that may affect the sensors. NOVOSENSE recommends using anti-static, sealable bags for storage. Do not use adhesive or tape in storage containers.

3.3.Reflow

NSHT30 can be soldered using a standard reflow soldering furnace. It can withstand the welding process, including the curve in accordance with IPC/JEDEC J-STD-020, with a peak temperature of 260 °C. According to the IPC/JEDEC J-STD-020 standard, the classification temperature is specified based on the packaging thickness and size. The lead-free process classification temperature (Tc) table in the IPC/JEDEC J-STD-020 standard lists these temperatures, as shown in Table 3.1.

Table 3.1 Different lead-free process classification temperatures

Package Thickness	Volume＜350mm3	Volume＜350-2000mm3	Volume＞2000mm3
＜1.6mm	260℃	260℃	260℃
1.6mm-2.5mm	260℃	260℃	245℃
＞2.5mm	250℃	245℃	245℃

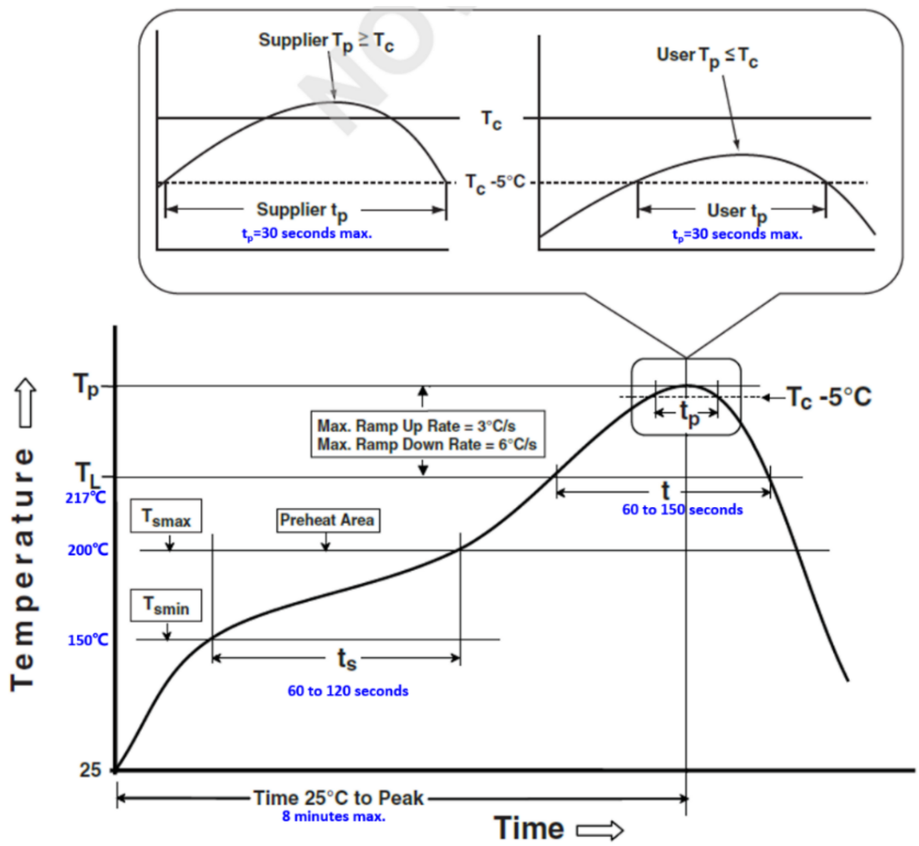


Figure 3.2 The Reflow Curve

Figure 3.2 contains the key temperatures and times for each temperature zone of reflow soldering. The peak temperature (Tp) of the reflow curve is measured at the top of the sensor package.
-For IC supplier: Tp≥Tc;
-For user: Tp≤Tc;
- The time (tp) for defining the peak temperature starts/ends at Tc-5℃.
When soldering the NSHT30, it is necessary to ensure compliance with the maximum temperature and exposure time. If the PCB needs multiple soldering cycles, we recommend soldering the temperature and humidity sensors in the last cycle to reduce the risk of sensor damage. It is strongly recommended to use solder paste with a "no cleaning" tin powder particle size of T3 or above, because it does not require washing the board, which may be harmful to the sensor.

It should be noted that the NSHT30 should be reflow less than three times, which can reduce the problem of bad contact. Because the top of the NSHT30 package has an open hole structure, after reflow, the holes of the sensor can be covered with high-temperature tape before soldering. It can prevent flux entering the holes which can affect the accuracy of humidity.

After reflow process, the humidity sensors may temporarily get a negative humidity offset. This is because when a relative humidity sensor is exposed to the high temperature in the soldering process, the sensor element tends to dry out. The offset will slowly disappear by itself when the sensor is exposed to ambient conditions.
To avoid the offset in the relative humidity readings, the sensor element must be rehydrated after the soldering process. The recommended rehydration conditions are either:
-A relative humidity > 75% RH at room temperature for at least 12 hours
-A relative humidity of 40% ~ 50% RH at room temperature for 3 to 5 days

3.4.Avoid Chemical

The NSHT30 must be protected from exposure to volatile chemicals, organic compounds, acids, alkalis, cleaning agents, ketones, ethanol, isopropanol, toluene, adhesives, and other environments to avoid irreversible effects on the humidity accuracy.

4.Demo Board Design Reference

4.1.NSHT30 Hardware Design

4.1.1.DEMO schematic

The hardware of demo consists of two parts: one is a collection and communication module based on a 51 microcontroller, and the other is a NSHT30 module, which is connected to the two modules through a ribbon cable. The circuit schematic and layout are shown in the following figure.

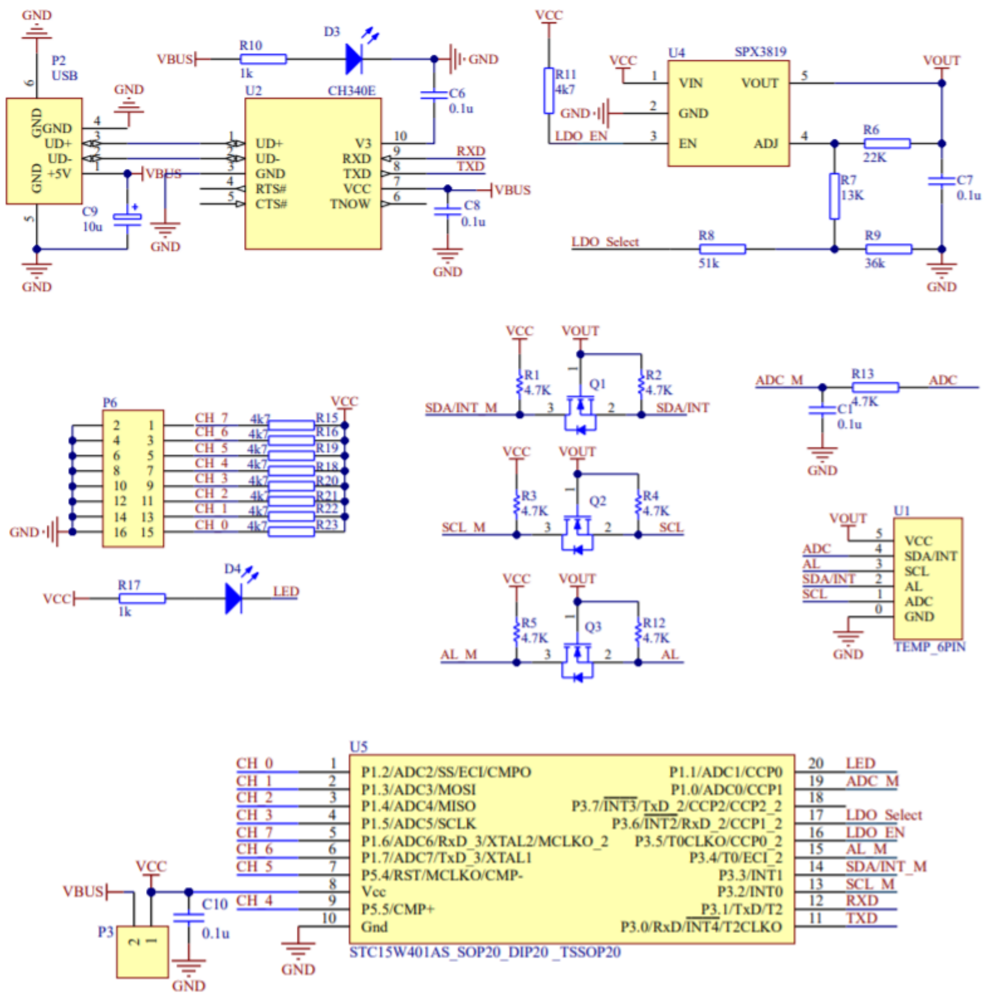


Figure 4.1 Communication Module

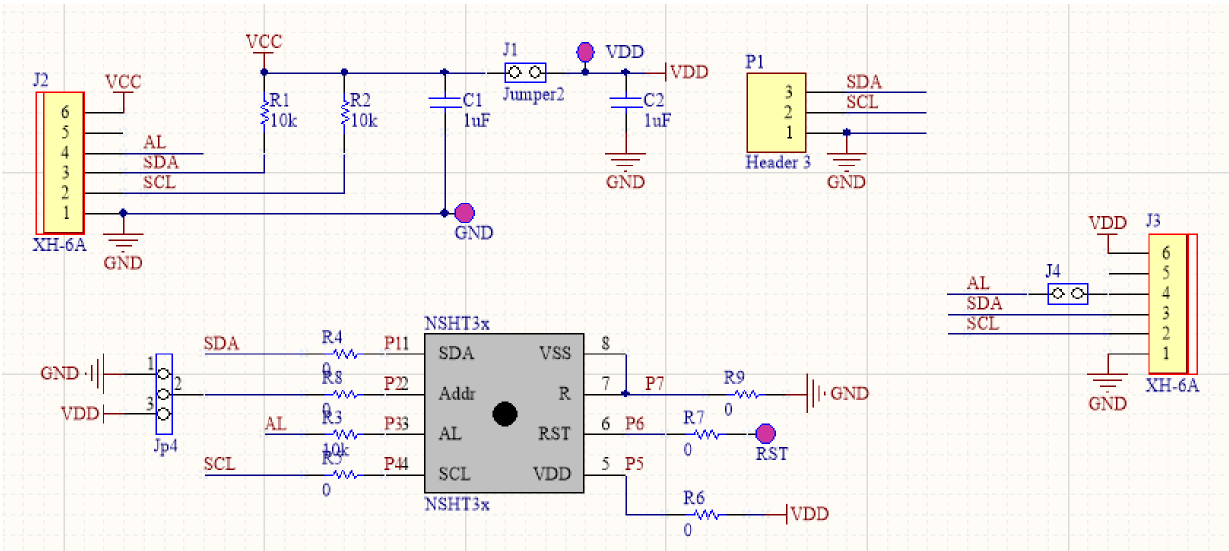


Figure 4.2 NSHT30 application circuit

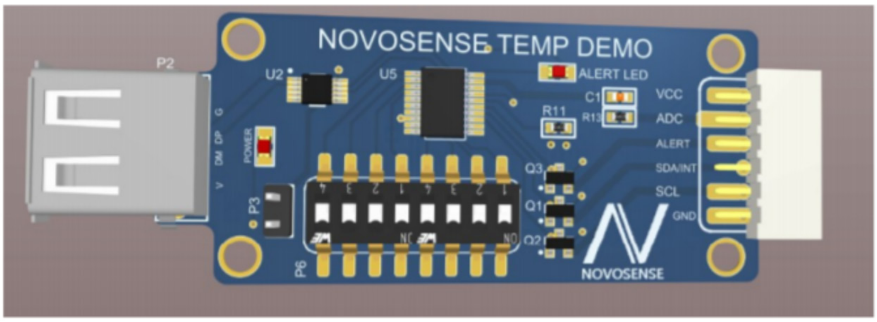


Figure 4.3 Collection and Communication Module based on a 51 microcontroller

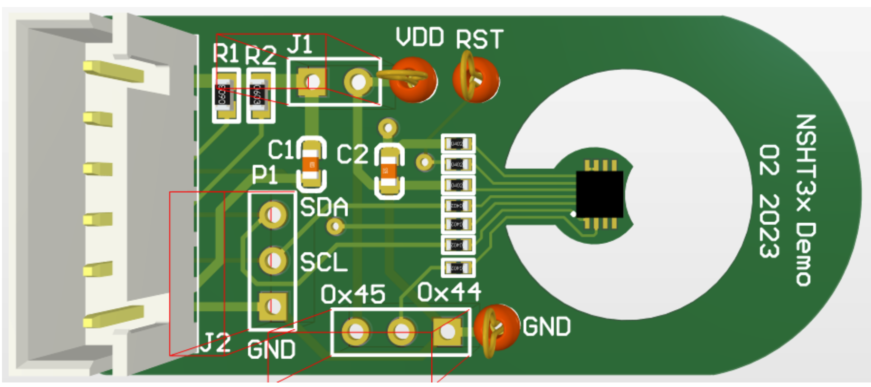


Figure 4.4 NSHT30- Soldering PCB

4.1.2.Layout guidance

1. The NSHT30 should be positioned as far away from heat sources as possible.
2. Decrease the heat conduction between theNSHT30 and the PCB. For example, don't covered with copper around the sensor. The wiring should be as thin as possible. The holes should be made around the sensor on the PCB which can reduce heat conduction. Alternatively, a flexible soft board can be used to physically isolate the NSHT30 from the test board.
3. Reduce the heat transfer between the NSHT30 and the PCB. The PCB should be as thin as possible, such as using a separate FPC soft board to assemble the temperature and humidity sensor.

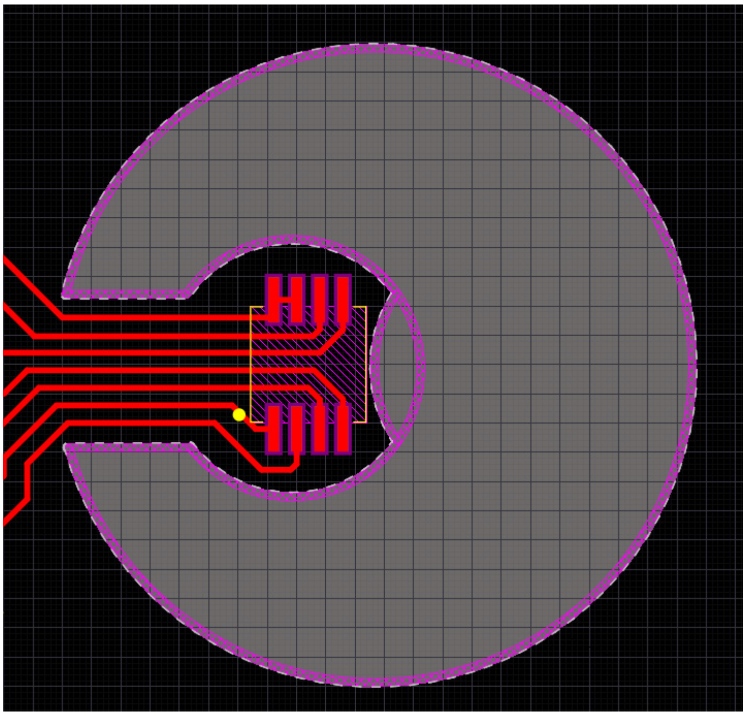


Figure 4.5 Layout Sample

4.2.NSHT30 Driver Code

```
*********************************************************************************
// IIC Driver Code
void IIC_Start(void)
{
        SDA_OUT();
        IIC_SDA=1;
        IIC_SCL=1;
        delay_us(1);
        IIC_SDA=0;//START:when CLK is high,DATA change form high to low
        delay_us(1);
        IIC_SCL=0;
}
void IIC_Stop(void)
{
        SDA_OUT();
        IIC_SCL=0;
        IIC_SDA=0;//STOP:when CLK is high DATA change form low to high
        delay_us(1);
        IIC_SCL=1;
        delay_us(1);
        IIC_SDA=1;
        delay_us(1);
}
uint8_t IIC_Wait_Ack(void)
{
        uint32_t ucErrTime=0;
        SDA_IN();
        IIC_SDA=1;delay_us(1);
        IIC_SCL=1;delay_us(1);
        while(READ_SDA)
        {
                ucErrTime++;
                if(ucErrTime>1000)
                {
                        IIC_Stop();
                        return 1;
                }
        }
        IIC_SCL=0;
```

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

void IIC_Ack(void)
{
        SDA_OUT();
        IIC_SCL=0;
        delay_us(2);
        IIC_SDA=0;
        delay_us(2);
        IIC_SCL=1;
        delay_us(2);
        IIC_SCL=0;
        delay_us(2);
        IIC_SDA=1;
}
void IIC_NAck(void)
{
        SDA_OUT();
        IIC_SCL=0;
        delay_us(2);
        IIC_SDA=1;
        delay_us(2);
        IIC_SCL=1;
        delay_us(2);
        IIC_SCL=0;
        delay_us(2);
}
void IIC_Send_Byte(uint8_t txd)
{
        uint8_t t;
        SDA_OUT();
        IIC_SCL=0;
  for(t=0;t<8;t++)
  {
        IIC_SDA=(txd&0x80)>>7;
        txd<<=1;
        delay_us(2);
```

```
            IIC_SCL=1;
            delay_us(2);
IIC_SCL=0;
            delay_us(2);
  }
}
uint8_t IIC_Read_Byte()
{
            unsigned char i,receive=0;
            SDA_IN();
            for(i=0;i<8;i++ )
            {
                     IIC_SCL=0;
                     delay_us(2);
IIC_SCL=1;
                     receive<<=1;
                    if(READ_SDA)
receive++;
delay_us(2);
            }
            IIC_SCL=0;
            return receive;
}
//NSHT30 Driver Code
#define  CMD_MEAS_SINGLE_H  0x2400 // measurement: SINGLE Mode high repeatability
#define  CMD_MEAS_SINGLE_M  0x240B // measurement: SINGLE Mode  medium repeatability
#define  CMD_MEAS_SINGLE_L  0x2416 // measurement: SINGLE Mode  low repeatability
#define  CMD_MEAS_PERI_05_H  0x2032 // measurement: periodic Mode 0.5 mps high repeatability
#define  CMD_MEAS_PERI_05_M  0x2024 // measurement: periodic Mode 0.5 mps medium repeatability
#define  CMD_MEAS_PERI_05_L  0x202F // measurement: periodic Mode 0.5 mps low repeatability
#define  CMD_MEAS_PERI_1_H   0x2130 // measurement: periodic Mode 1 mps high repeatability
#define  CMD_MEAS_PERI_1_M  0x2126 // measurement: periodic Mode 1 mps medium repeatability
#define  CMD_MEAS_PERI_1_L   0x212D // measurement: periodic Mode 1 mps low repeatability
#define  CMD_MEAS_PERI_2_H   0x2236 // measurement: periodic Mode 2 mps high repeatability
#define  CMD_MEAS_PERI_2_M  0x2220 // measurement: periodic Mode 2 mps medium repeatability
#define  CMD_MEAS_PERI_2_L   0x222B // measurement: periodic Mode 2 mps low repeatability
#define  CMD_MEAS_PERI_4_H   0x2334 // measurement: periodic Mode 4 mps high repeatability
#define  CMD_MEAS_PERI_4_M  0x2322 // measurement: periodic Mode 4 mps medium repeatability
#define  CMD_MEAS_PERI_4_L   0x2329 // measurement: periodic Mode 4 mps low repeatability
#define  CMD_MEAS_PERI_10_H  0x2737 // measurement: periodic Mode 10 mps high repeatability
```

```
#define  CMD_MEAS_PERI_10_M  0x2721 // measurement: periodic Mode 10 mps medium repeatability
#define  CMD_MEAS_PERI_10_L  0x272A // measurement: periodic Mode 10 mps low repeatability
void nsht30_set_periodic(uint8_t addr, uint16_t cmd)
{
        IIC_Start();
        IIC_Send_Byte(addr<<1);
        IIC_Wait_Ack();
        IIC_Send_Byte((cmd & 0xFF00) >> 8);
        IIC_Wait_Ack();
        IIC_Send_Byte(cmd & 0xFF);
        IIC_Wait_Ack();
        IIC_Stop();
}
//Periodic Mode
void nsht30_read_raw_periodic(uint8_t addr,uint8_t *buff)
{
        IIC_Start();
        IIC_Send_Byte(addr<<1);
        IIC_Wait_Ack();
        IIC_Send_Byte(0xE0);//Read cmd 0xE000
        IIC_Wait_Ack();
        IIC_Send_Byte(0x00);
        IIC_Wait_Ack();
        delay_ms(10);
        IIC_Start();
        IIC_Send_Byte((addr<<1)+1);
        if(IIC_Wait_Ack()==0)
         {
                buff[0]=IIC_Read_Byte();
                IIC_Ack();
                buff[1]=IIC_Read_Byte();
                IIC_Ack();
                buff[2]=IIC_Read_Byte();
                IIC_Ack();
                buff[3]=IIC_Read_Byte();
                IIC_Ack();
                buff[4]=IIC_Read_Byte();
                IIC_Ack();
                buff[5]=IIC_Read_Byte();
                IIC_NAck();
```

```
                IIC_Stop();
        }
}
//Single Mode
void nsht30_read_raw_single(uint8_t addr, uint16_t cmd, uint8_t *buff)
{
                IIC_Start();
                IIC_Send_Byte(addr<<1);
                IIC_Wait_Ack();
                IIC_Send_Byte((cmd & 0xFF00) >> 8);
                IIC_Wait_Ack();
                IIC_Send_Byte(cmd & 0xFF);
                IIC_Wait_Ack();
                IIC_Stop();
                delay_ms(10);
                IIC_Start();
                IIC_Send_Byte((addr<<1)+1);
 if(IIC_Wait_Ack()==0)
        {
                buff[0]=IIC_Read_Byte();
                IIC_Ack();
                buff[1]=IIC_Read_Byte();
                IIC_Ack();
                buff[2]=IIC_Read_Byte();
                IIC_Ack();
                buff[3]=IIC_Read_Byte();
                IIC_Ack();
                buff[4]=IIC_Read_Byte();
                IIC_Ack();
                buff[5]=IIC_Read_Byte();
                IIC_NAck();

                IIC_Stop();
        }
}


//CRC calculation
int nst3x_crc_check(uint8_t *data, uint8_t len, uint8_t checksum)
{
```

```
            uint8_t crc = 0xFF, bit;
            uint8_t byteCtr;
    //calculates 8-Bit checksum with given polynomial
            for (byteCtr = 0; byteCtr < len; ++byteCtr)
    {
            crc ^= (data[byteCtr]);
            for (bit = 8; bit > 0; --bit)
      {
            if (crc & 0x80)
            crc = (crc << 1) ^ 0x131;
            else
            crc = (crc << 1);
      }
    }
    if (crc == checksum)
      return 1;
    else
      return 0;
}
//Reading and calculating temperature and humidity
HAL_StatusTypeDef read_temp_rh_1ch(uint8_t addr, double *pout)
{
            uint8_t dat[6];
            uint16_t tem,hum;
            nsht30_read_raw_single(addr,CMD_MEAS_SINGLE_L,dat);// single mode
            tem = ((uint16_t)dat[0]<<8) | dat[1];
            hum = ((uint16_t)dat[3]<<8) | dat[4];
             if((nst3x_crc_check(dat,2,dat[2])) && (nst3x_crc_check(dat+3,2,dat[5])))
                        {
                                    pout[0]= (175.0*(double)tem/65535.0-45.0) ;// T = -45 + 175 * tem / (2^16-1)
                                    pout[1]= (100.0*(double)hum/65535.0);// RH = hum*100 / (2^16-1)
                                     return HAL_OK;
                         }
                        else
                         {
                                    return HAL_ERROR;
                        }
}
//main
double rth[2];
```

```
int main(void)
{
        uint8_t addr = 0x44;
        MX_GPIO_Init();
        MX_USART2_UART_Init();
        IIC_Init();
        while (1)// cycling test per second
        {
                delay_ms(1000);
                if(read_temp_rh_1ch(addr,rth) == HAL_OK)
                {
                        printf("%3.4f,%3.6f%%\r\n",rth[0],rth[1]);
                }
                else
                {
                        printf("crc error\r\n");
                }
 }
```

Part of the NSHT30 driver code can be found in the above code, Part ID recognition is not required after power on.
For further information , please contact NOVOSENSE.

5.Assemble the Sensor

In order to fully contact the sensor with the testing environment, it is recommended to follow several basic principles during assembly:

1.There must be a contact channel between the temperature and humidity sensor and the testing environment;

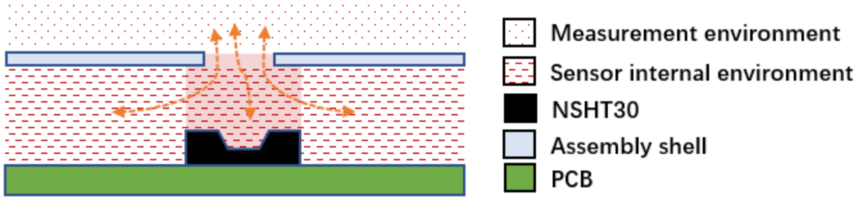


Figure 5.1 Good contact between sensor and environment

2.Set physically isolate between the sensor area and the inside area, it can improve response speed and reduce the impact of air inside area on the relative humidity value;

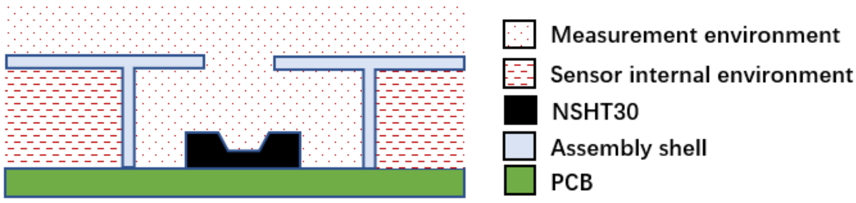


Figure 5.2 Physically Isolate between the sensor and the inside area

3. Reduce the volume of the area formed between the sensor and the shell, further improving the response time of the testing environment;

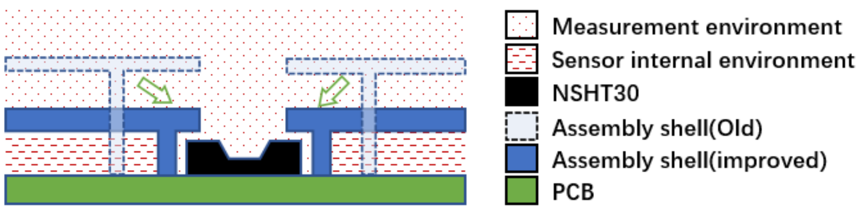


Figure 5.3 Reduce the volume of the area between the sensor and the shell

4. If the application is harsh and there is a risk of dust or water splashing, a waterproof layer can be added at the opening of the shell.

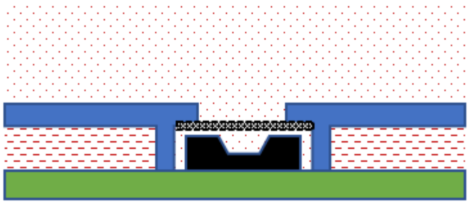


Figure 5.4 Add the Waterproof

In addition, there are several basic principles;
Relative Humidity

$$RH = \frac{\text{water vapor content in the air}}{\text{The maximum water vapor content that air can absorb}}$$

The sensor is in full contact with the test environment, and the content of water vapor in the air can be basically consistent between the sensor and the test environment; However, the maximum concentration of water vapor that air can absorb is related to air temperature. The higher the temperature, the higher the maximum concentration of water vapor that air can absorb; Therefore, only when the temperature measured by the sensor is consistent with the temperature of the measurement environment, the relative humidity value is the true relative humidity value of the monitored area.

To ensure that the sensor temperature is consistent with the ambient temperature to be measured, propose the following three directions:
1. The sensor should be located as close as possible to the area to be measured and avoid the heat sources .It can avoid the inconsistency between the relative humidity of the sensor area and the measurement area caused by the heating of the air in the test area by the heat source;

2. There are several basic methods to reduce the thermal inertia of the sensor area:

1)Reduce the PCB thickness;

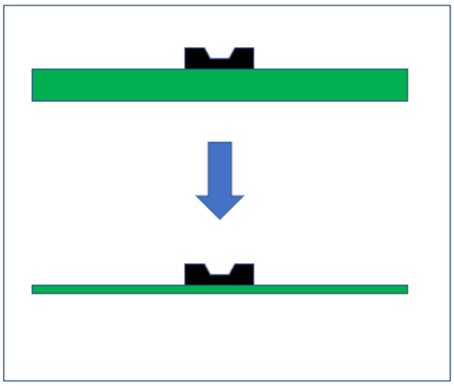


Figure 5.5 Reduce the PCB thickness

2)Use an independent small board, connected to the motherboard through pins;

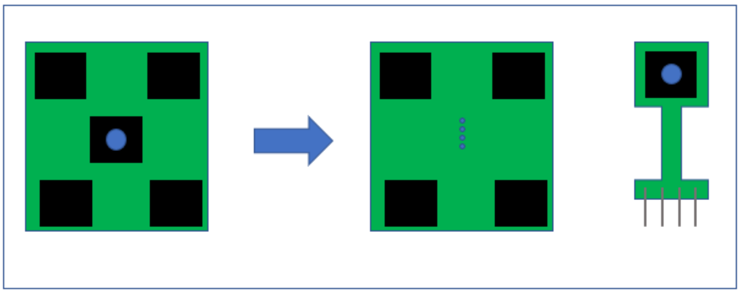


Figure 5.6 Use an independent small board

3)Use FPC soft board;

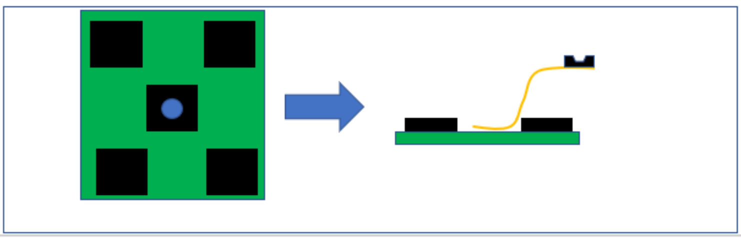


Figure 5.7 Use FPC soft board

3. Reducing the impact of other areas to the sensor’ s temperature

1)Set physically isolate between the sensor and the board; Reduce the impact of air heat conduction and radiation;

2)Avoid direct sunlight and reduce the temperature increase caused by the heat transmitted through the thermal radiation path;

3)Sensors should be installed independently on the board as much as possible; If it must be installed on the motherboard, the sensor’ s position should be as far away from the heat source as possible; On the board, holes should be opened near the sensor; Reduce the impact of heat conduction on sensor temperature caused by heat sources passing through the circuit board;

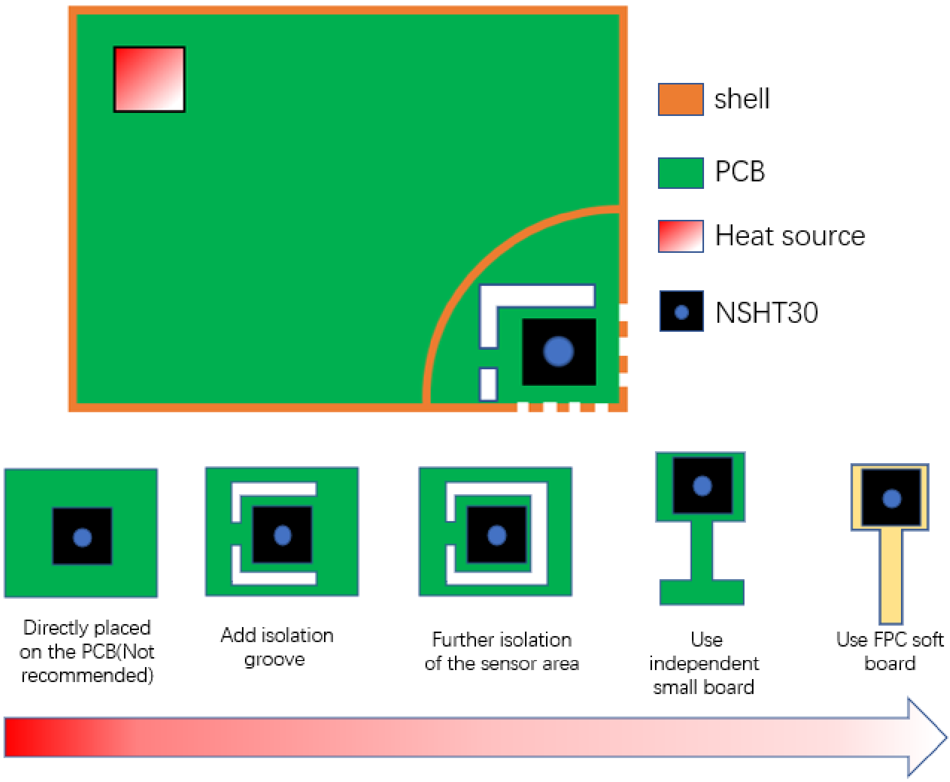


Figure 5.8 Degree of influence from circuit board heat conduction

For more product information, please visit www.novosns.com

To request samples, please send an email to sales@novosns.com

6.Revision History

Revision	Description	Author	Date
1.0	Initial version	Juanjuan Shao	2023/10/30