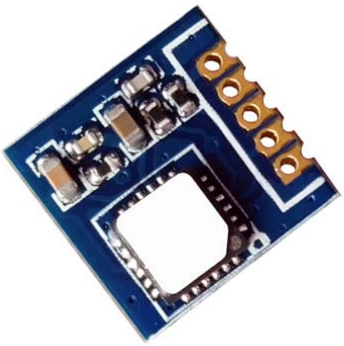
DSTH01
Digital Temperature Humidity Sensor Module V1.00

Features:

- I2C host interface
- Temperature range: -40°C ~+85°C
- Temperature accuracy: ±0.5°C (typical)
  ±1°C (Max. @0~70°C)
- RH operating range: 0~100%
- RH accuracy: ±3% (typical)
- Wake up time: 10ms
- Operating voltage range: 2.1~3.6V
- Integrated on-chip heater
- Low power consumption
- Excellent long term stability
- Factory calibrated



Applications

- Industrial HVAC/R
- Thermostats / humidistats
- Respiratory therapy
- Automotive climate control
- Asset and goods tracking

DESCRIPTION

DSTH01 is a type of digital relative humidity and temperature sensor module which integrates temperature and humidity sensor elements, an analog-to-digital converter, signal processing, calibration data and an I2C host interface. Both the temperature and humidity sensors are factory-calibrated and the calibration data is stored in the on-chip non-volatile memory which ensures the DSTH01 modules are fully interchangeable and no recalibration or software changes are required.

Necessary components are integrated on the DSTH01 modules so users can get a quick start with the microcontroller with extra design. The sensor top is covered with silicon gel which can protect the sensor from the dust and other particles. The module works at 2.1~3.6V. It consumes about 240uA during RH conversion. and 1.5mA in normal work mode. The DSTH01 module offers an accurate and factory-calibrated solution for embedded applications ranging from HVAC/R system to consumer electronic products.

PIN FUNCTIONS

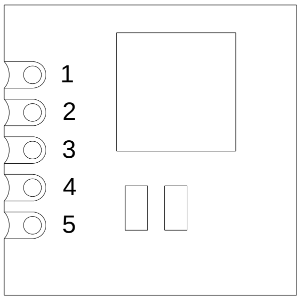


Figure 1: DSTH01 Pin Layout

PIN	DIP	Function	Description
1	GND	Ground	Ground (0V)
2	SCK	Input	I2C serial port, clock pin
3	SDA	Input/output	I2C serial port, data pin
4	/CS	Input	Chip selection, low effective
5	VCC	Power	Power supply

Table 1: DSTH01 Pin functions

ELECTRICAL SPECIFICATIONS

Symbol	Parameter (condition)	Min.	Typ.	Max.	Units
VCC	Supply Voltage	2.1	3.3	3.6	V
Temp	Temperature operating range	-40	25	85	°C
T_R(1)	Temperature resolution			14	bit
				1/32	°C
T_A(2)	Temperature accuracy: typical @25°C		±0.5	±1	°C
	Maximum	See figure 2			°C
T_{RN}	Temperature repeatability-noise			0.1	°C RMS
T_{RT}(3)	Response time to reach 63% of final value		1.5		s
T_{LS}	Temperature long term stability		<0.05		°C/yr
RH(4)	Relative humidity operating range	0		100	%
H_R(5)	Humidity resolution			12	bit
H_A(6)	Humidity accuracy: 20~80%RH		±3	±4.5	%RH
	0~100%RH	See figure 3			%RH
H_{RN}	Humidity repeatability-noise		0.05		%RH RMS

H_{RT}(7)	Response time @ 1m/s airflow		8		s
H_H	Hysteresis		±1		%RH
H_{LS}	Humidity long term stability		≤0.25		%RH
I_{DD}	Current. @ RH conversion in progress		240	560	uA
	@ Temp conversion in progress		320	565	uA
	@ Heater enabled, on conversion in progress		24	31	mA
T_{CON}	Conversion time. @ 14-bit temp, 12-bit RH (fast=0)		35	40	ms
	@ 13-bit temp, 11-bit RH (fast=1)		18	21	
T_{PU}	Power up time		10	15	ms

Table 2: DSTH01 Electrical Specifications

Notes:

(1). The DSTH01 module has a nominal output of 32 codes /°C, with 0000=-50°C

(2). Temperature sensor accuracy is for VDD = 2.3 to 3.6 V.

(3). Actual response times will vary dependent on system thermal mass and air-flow.

(4). Recommended humidity operating range is 20 to 80%RH (non-condensing) over 0 to 60℃. Prolonged operation beyond these ranges may result in a shift of sensor reading, with slow recovery time.

(5). The DSTH01 module has a nominal output of 16 codes per %RH, with 0h0000=-24%RH.

(6). Excludes hysteresis, long-term drift, and certain other factors and is applicable to non-condensing environments only.

(7). Time for sensor output to reach 63% of its final value after a step change.

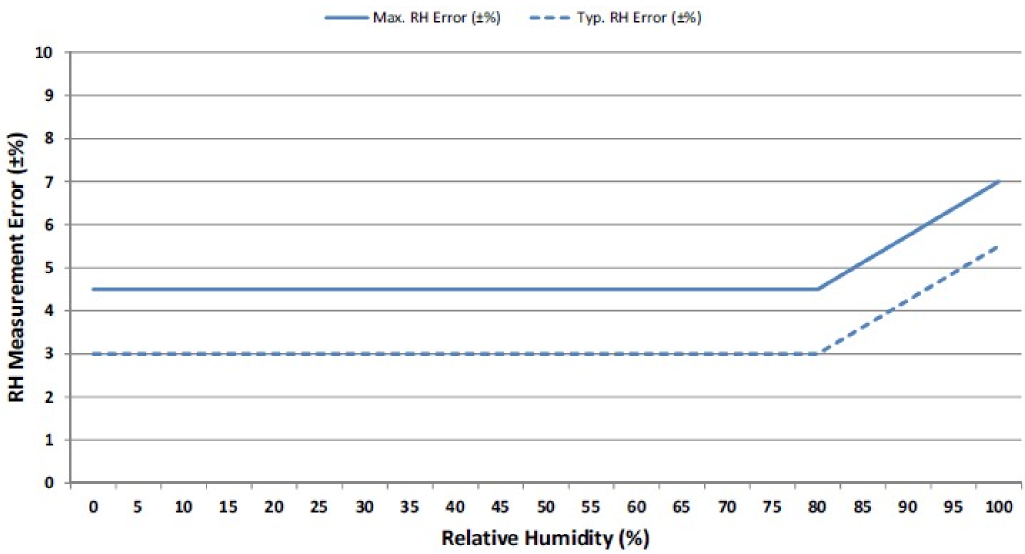


Figure 2: DSTH01 RH Accuracy at 30°C

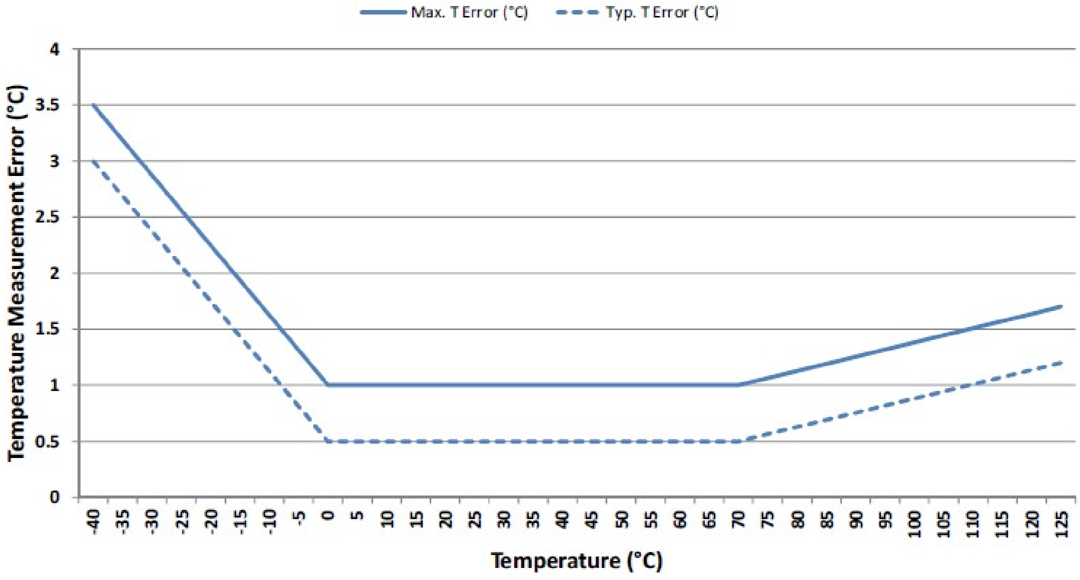


Figure 3: DSTH01 Temperature Accuracy

Symbol	Parameter (condition)	Min.	Typ.	Max.	Units
V_{HYS}	Hysteresis	0.05*VCC			V
F_{SCL}	SCLK frequency			400	kHz
T_{SKH}	SCL high time	0.6			us
T_{SKL}	SCL low time	1.3			us
T_{STH}	Start hold time	0.6			us
T_{STS}	Start setup time	0.6			us
T_{SPS}	Stop setup time	0.6			us
T_{BUS}	Bus free time between stop and start	1.3			us
T_{DS}	SDA setup time	100			ns
T_{DH}	SDA hold time	100			ns
T_{VD}	SDA valid time			0.9	us
T_{VT}	SDA acknowledge time			0.9	us

Table 3: I^2C Interface Specifications

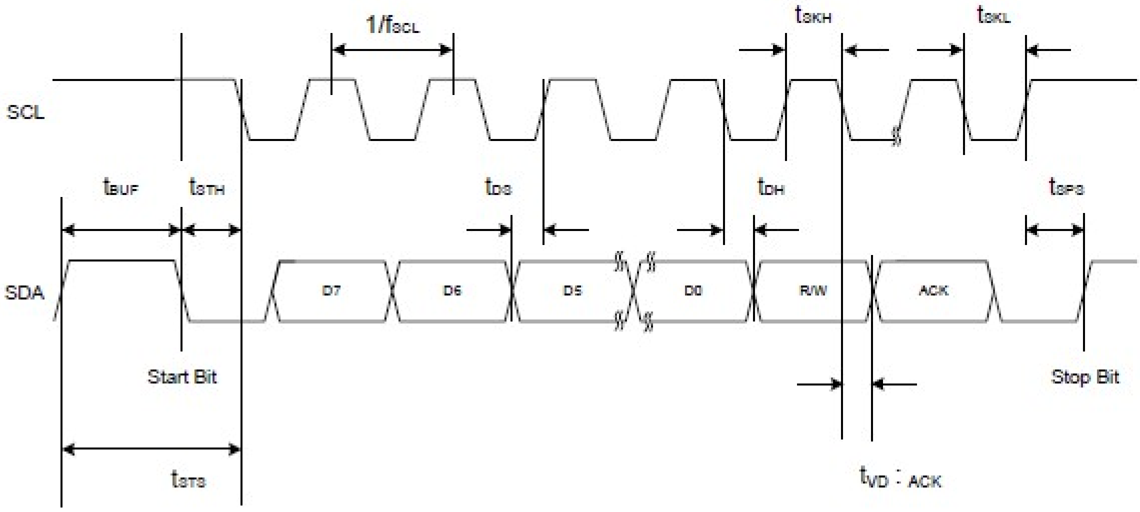


Figure 4: I²C Interface Timing Diagram

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Min.	Max.	Units
V_D	Voltage on VCC with respect to GND	-0.3	4.2	V
V_{I2C}	Voltage on SDA or SCL pin with respect to GND	-0.3	3.9	V
T_{AT}(1)	Ambient temperature under bias	-55	125	°C
T_{ST}	Storage temperature	-55	150	°C

Table 4: DSTH01 Maximum Ratings

Notes: For best accuracy, the DSTH01 module should be stored in climate controlled conditions (10 to 35°C, 20 to 60%RH). Exposure to high temperature and/or high humidity environments can cause a small upwards shift in RH readings.

HOST INTERFACE

1. I²C Interface

The DSTH01 sensor module has an I2C serial interface with a 7-bit address of 0x40. It is a slave device supporting data transfer with rates up to 400 kHz. The table 5 shows the register summary of the DSTH01 modules. Users can configure related values to obtain corresponding temperature and humidity parameters.

Register	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	STATUS	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	/RDY
0x01	DATAh	Relative humidity or temperature, High byte							
0x02	DATAl	Relative humidity or temperature, Low byte							
0x03	CONFIG	RSVD	RSVD	FAST	TEMP	RSVD	RSVD	HEAT	START

0x11	ID	ID3	ID2	ID1	ID0	0	0	0	0

Table 5: Summary of Registers

Please note that any register address which is not listed here is reserved and must not be written. The Reserved Register Bits (RSVD) must always written as zero; the result of a read operation on these bits is undefined.

2. Performing a Relative Humidity Measurement

The following steps must be executed in sequence in order to take a relative humidity measurement.

(1). Set START bit (bit0) and clear TEMP bit (bit4) in CONFIG register (0x03) to begin a new conversion, i.e.: write CONFIG (0x03) with value 0x01

(2). Poll RDY(D0) in STATUS register (0x00) until it is low (=0).

(3). Read the upper and lower bytes of the RH value from DATAh and DATAL registers (0x01 & 0x02) respectively.

DATAh								DATAl							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
12-bit relative humidity code											0	0	0	0	0

Table 6: 12-bit Relative Humidity Result in Registers DATAh and DATAl

(4). Convert the RH value to %RH with the equation: **%RH = (RH/16)-24**, which the RH is the measured value returned in DATAh & DATAl.

(5). Apply temperature compensation and /or linearization which will be discussed in the following section. The table below shows the values that corresponding to various measured RH levels.

%RH	12 Bit Code	
	Dec	Hex
0	384	180
10	544	220
20	704	2C0
30	864	360
40	1024	400
50	1184	4A0
60	1344	540
70	1504	5E0
80	1664	680
90	1824	720
100	1984	7C0

Table 7: Typical %RH Measurement Codes for 0 to 100% RH Range

The above sequence assumes to be in normal mode, Tconv=35ms (typical). Conversion also can be performed in fast mode.

3. Performing a Temperature Measurement

(1). Set START bit (bit0) and TEMP bit (bit4) in CONFIG register (0x03) to begin a new conversion, i.e.: write CONFIG (0x03) with value 0x11

(2). Poll RDY(D0) in STATUS register (0x00) until it is low (=0).

(3). Read the upper and lower bytes of the RH value from DATAh and DATAL registers (0x01 & 0x02) respectively.

DATAh								DATAl							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
12-bit relative humidity code											0	0	0	0	0

Table 8: 14-bit Temperature Result in Registers DATAh and DATAl

(4). Convert the value to temperature with the equation: **Temperature(°C) = (TEMP/32)-50**, which the TEMP is the measured value returned in DATAh & DATAl.

TEMP(°C)	14 Bit Code	
	Dec	Hex
-40	320	0140
-30	640	0280
-20	960	03C0
-10	1280	0500
0	1600	0640
10	1920	0780
20	2240	08C0
30	2560	0A00
40	2880	0B40
50	3200	0C80
60	3520	0DC0
70	3840	0F00
80	4160	1040
90	4480	1180
100	4800	12C0

Table 9: Typical Temperature Measurement Codes for -40°C to 100°C Range

4. Normal Conversion Mode and Fast Conversion Mode

The switch between two modes is realized by setting the value of FAST (bit5) in CONFIG register (0x03). Fast=0 is normal mode and Fast=1 is fast mode.

Mode	T_{CON} (Typical)	Temperature Resolution	Humidity resolution
Normal Mode	35ms	14-bit	12-bit
Fast Mode	18ms	13-bit	11-bit

Table 10: Normal Conversion Mode vs Fast Conversion Mode

5. Heater

The sensor chip on DSTH01 module integrates a resistive heating element which may be used to raise the temperature of the humidity sensor. This element can be used to drive off condensation or to implement dew-point measurement when the module is used in conjunction with a separate temperature sensor such as another DSTH01 module.

The heater can be activated by setting HEAT (D1) in CONFIG (register 0x03). Turning on the heater will reduce the tendency of the humidity sensor to accumulate an offset due to "memory" of sustained high humidity conditions. When the heater is enabled, the reading of the on-chip temperature sensor will be affected (increased).

6. I^{2}C Operation

If the DSTH01 module shares the I^{2}C bus with other slave devices, it should be powered down when the master controller is communicating with the other slave devices, which can be realized either by setting /CS to logic high or setting the VCC pin to 0V. User can power off the module by using GPIO pin to control the VCC of DSTH01 module. Please note that users must consider the driving current of GPIO when using the heater function of the module, in which function the current of the enabled heater might consumes the current more than 30mA (seeing the table of ELECTRIC SPECIFICATIONS).

A6	A5	A4	A3	A2	A1	A0	R/W
1	0	0	0	0	0	0	1/0

Table 11: I^{2}C Slave Address Byte

(1). I^{2}C write operation

To write a register on DSTH01 module, the master should issue a start command (S) followed by the slave address-0x40. The slave address should be followed by a 0 to indicate that the operation is a write. Upon recognizing its slave address, the DSTH01 issues an acknowledge (A) by pulling the SDA line low for the high duration of the ninth SCL cycle. The next bye which the master places on the bus is the register address pointer, selecting the register on the DSTH01 to which the data should be transferred. After the DSTH01 acknowledges this byte, the master places a data byte on the bus. This byte will be written to the register selected by the address pointer. The DSTH01 module will acknowledge the data byte, after which the master issues a Stop command (P).

Master	Slave

Table 12: Identification for Master and Slave Data

S	Slave Address	W	A	Address Pointer	A	Register Data	A	P

Table 13: Sequence to Write a Register

S	0x40	0	A	0x03	A	0x01	A	P

Table 14: Sequence to Start a Relative Humidity Conversion

S	0x40	0	A	0x03	A	0x11	A	P

Table 15: Sequence to Start a Temperature Conversion

(2). I^{2}C read operation

To read a register on the DSTH01 module, the master must first set the address pointer to indicate the register from which the data is to be transferred. The master should issue a start command (S) followed by the slave address---0x40. The slave address is followed by a 0 to indicate that the operation is a write. Upon recognizing its slave address, the DSTH01 will issue an acknowledge (A) by pulling the SDA line low for the high duration of the ninth SCL cycle. The next byte the master places on the bus is the register address pointer selecting the register on the DSTH01 from which the data should be transferred. After the DSTH01 acknowledges this byte, the master issues a repeated start command (Sr) indicating that a new transfer is to take place. The DSTH01 is addressed once again with the R/W bit set to 1, indicating a read operation. The DSTH01 will acknowledge its slave address and output data from the previously-selected register onto the data bus under the control of the SCL signal, the master should not acknowledge ($\overline{A}$) the data byte and issue a stop (P) command. However, if a RH or Temperature conversion result (two bytes) is to be read, the master should acknowledge (A) the first data byte and continue to activate the SCL signal. The DSTH01 will automatically output the second data byte. Upon receiving the second byte, the master should issue a not Acknowledge ($\overline{A}$) followed by a stop command.

S	Slave Address	W	A	Address Pointer	A	Sr	Slave Address	R	A	Register Data	$\overline{A}$	P

Table 16: Sequence to Read from a Single Register

S	0x40	0	A	0x11	A	Sr	0x40	1	A	ID	$\overline{A}$	P

Table 17: Sequence to Read Device ID

S	0x40	0	A	0x11	A	Sr	0x40	1	A	---	$\overline{RDY}$	$\overline{A}$	P

Table 18: Sequence to Read $\overline{RDY}$ Bit

S	Slave Address	W	A	Address Pointer	A	Sr	Slave Address	R	A	Register1 Data	A	Register2 Data	$\overline{A}$	P
S	0x40	0	A	0x01	A	Sr	0x40	1	A	Data H	A	Data L	$\overline{A}$	P

Table 19: Sequence to Read Conversion Result

APPLICATION CODES

This section shows the basic communication between DSTH01 and STM8 microcontroller through I2C interface. Reading temperature and relative humidity parameters are demonstrated in the codes.

```
#include "stm8l10x.h"

#define SDA_H        GPIO_SetBits(GPIOC, GPIO_Pin_0)
#define SDA_L        GPIO_ResetBits(GPIOC, GPIO_Pin_0)
#define SCK_H        GPIO_SetBits(GPIOC, GPIO_Pin_1)
#define SCK_L        GPIO_ResetBits(GPIOC, GPIO_Pin_1)
#define SDA          GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_0)
#define SlaveAddress    0x40
#define RegisterAddress0    0x00
#define RegisterAddress1    0x01
#define RegisterAddress2    0x02
#define RegisterAddress3    0x03
#define RegisterAddress11    0x11

void CLK_INT(void);
void GPIO_INT(void);
void DELAY(uint16_t n);
void I2C_START(void);
void I2C_STOP(void);
void I2C_WRITE(uint8_t Data);
uint8_t I2C_READ(void);
void I2C_ACK(uint8_t a);
uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
void Temperature_READ(uint8_t *s);
void RelativeHumidity_READ(uint8_t *s);

uint8_t Ack;
uint8_t id;
uint8_t RelativeHumidity;
uint8_t Temperature;
uint8_t Start1=0x01;
uint8_t Start2=0x11;
```

```
void main()
{
        CLK_INT();
        GPIO_INT();
        I2C_RECEIVE(SlaveAddress,RegisterAddress11,&id);

        while (1)
        {
            RelativeHumidity_READ(&RelativeHumidity);
            Temperature_READ(&Temperature);
        }
}

void CLK_INT()
{
        CLK_DeInit();
        DELAY(500);
        CLK_CCOCmd(ENABLE);
        CLK_MasterPrescalerConfig(CLK_MasterPrescaler_HSIDiv1);
}

void GPIO_INT()
{
        GPIO_DeInit(GPIOC);
        GPIO_Init(GPIOC,GPIO_Pin_1,GPIO_Mode_Out_PP_Low_Slow);
        GPIO_Init(GPIOC,GPIO_Pin_0,GPIO_Mode_Out_OD_Low_Slow);
}

void DELAY(uint16_t n)
{
        uint8_t i;
        while(n--)
        {
          for(i=0;i<16;i++);
        }
}

void I2C_START()
{
        SDA_H;
        SCK_H;
        SDA_L;
        SCK_L;
```

```
}

void I2C_STOP()
{
        SDA_L;
        SCK_H;
        SDA_H;
}

void I2C_WRITE(uint8_t Data)
{
        uint8_t i;
        for(i=0;i<8;i++)
        {
          if(Data&0x80)
            SDA_H;
          else
            SDA_L;
          SCK_H;
          SCK_L;
       Data<<=1;
        }
        SDA_H;
        SCK_H;
        if(SDA==1)
            Ack=0;
        else
            Ack=1;
        SCK_L;
}

uint8_t I2C_READ()
{
        uint8_t Data=0;
        uint8_t i;
        SDA_H;
        for(i=0;i<8;i++)
        {
          SCK_L;
          SCK_H;
          Data<<=1;
          if(SDA==1)
            Data=Data+1;
```

```
        }
        SCK_L;
        return Data;
}

void I2C_ACK(uint8_t a)
{
        if(a==0)
          SDA_L;
        else
          SDA_H;
        SCK_H;
        SCK_L;
}

uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
        SlaveAdd=SlaveAdd<<1;
        I2C_START();
        I2C_WRITE(SlaveAdd);
        if(Ack==0)
          return 0;
        I2C_WRITE(RegisterAdd);
        if(Ack==0)
          return 0;
          I2C_WRITE(*s);
          if(Ack==0)
            return 0;
        I2C_STOP();
        return 1;
}

uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
        SlaveAdd=SlaveAdd<<1;
        I2C_START();
        I2C_WRITE(SlaveAdd);
        if(Ack==0)
          return 0;
        I2C_WRITE(RegisterAdd);
        if(Ack==0)
          return 0;
        I2C_START();
```

```
        I2C_WRITE(SlaveAdd+1);
        if(Ack==0)
          return 0;
          *s=I2C_READ();
        I2C_ACK(1);
        I2C_STOP();
        return 1;
}

void RelativeHumidity_READ(uint8_t *s)
{
        uint8_t Status=1;
        uint8_t RelativeHumidityH;
        uint8_t RelativeHumidityL;
        uint16_t RelHum;
        I2C_SEND(SlaveAddress,RegisterAddress3,&Start1);
        while(Status==1)
        {
          I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
        }
        I2C_RECEIVE(SlaveAddress,RegisterAddress1,&RelativeHumidityH);
        I2C_RECEIVE(SlaveAddress,RegisterAddress2,&RelativeHumidityL);
        RelHum=RelativeHumidityH;
        RelHum=RelHum<<8;
        RelHum+=RelativeHumidityL;
        RelHum=RelHum>>4;
        *s=RelHum/16-24;
}

void Temperature_READ(uint8_t *s)
{
        uint8_t Status=1;
        uint8_t TemperatureH;
        uint8_t TemperatureL;
        uint16_t Temp;
        I2C_SEND(SlaveAddress,RegisterAddress3,&Start2);
        while(Status==1)
        {
          I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
        }
        I2C_RECEIVE(SlaveAddress,RegisterAddress1,&TemperatureH);
        I2C_RECEIVE(SlaveAddress,RegisterAddress2,&TemperatureL);
        Temp=TemperatureH;
```

```
		Temp<<=8;
		Temp+=TemperatureL;
		Temp>>=2;
		*s=Temp/32-50;
}
```

MECHANICAL DATA

Unit: mm

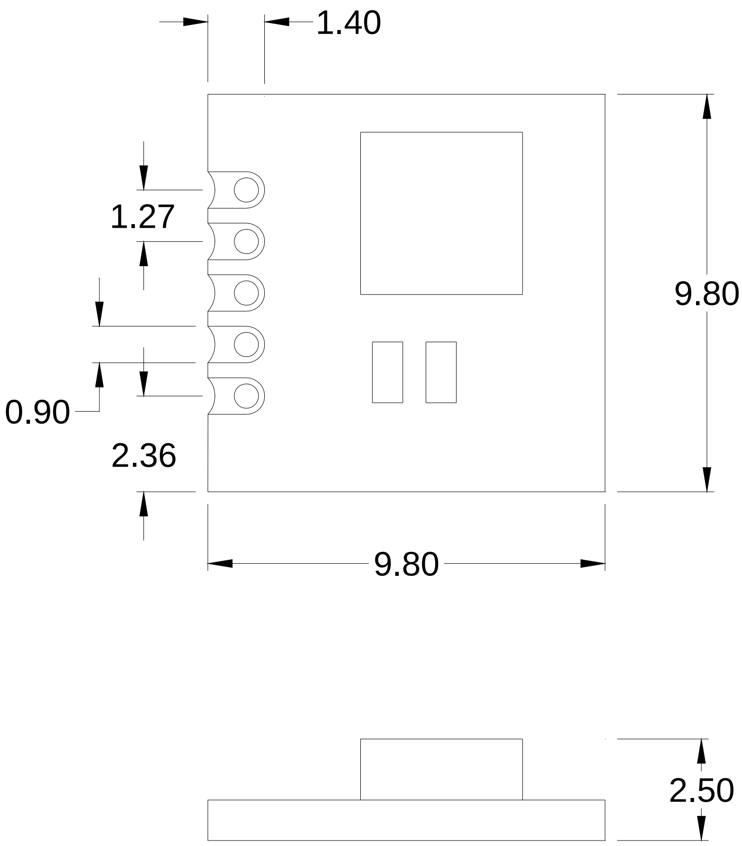


Figure 5: DSTH01 Dimensions

Dorji Applied Technologies A division of ***Dorji Industrial Group Co., Ltd*** Add.: Xinchenhuayuan 2, Dalangnanlu, Longhua, Baoan district, Shenzhen, China 518109 Tel: 0086-755-28156122 Fax.: 0086-755-28156133 Email: sales@dorji.com Web: http://www.dorji.com	Dorji Industrial Group Co., Ltd reserves the right to make corrections, modifications, improvements and other changes to its products and services at any time and to discontinue any product or service without notice. Customers are expected to visit websites for getting newest product information before placing orders. These products are not designed for use in life support appliances, devices or other products where malfunction of these products might result in personal injury. Customers using these products in such applications do so at their own risk and agree to fully indemnify Dorji Industrial Group for any damages resulting from improper use.