Infrared temperature sensor IR62

Manual V1.1

1. Product Overview

The infrared temperature sensor can calculate the surface temperature of the object by measuring the infrared radiation intensity emitted by the target without contacting the target. Non-contact temperature measurement is the biggest advantage of infrared thermometers, allowing users to conveniently measure inaccessible or moving targets.

IR62 online infrared thermometer is an integrated infrared temperature sensor, the sensor, optical system and electronic circuit are integrated in the stainless-steel shell;

The IR62 is easy to install, and the standard thread on the metal shell can be quickly connected to the installation site.

1.1 Technical parameter

Working power	6 ~ 32 VDC
Output signal	0~5V/0~10V/4~20mA/RS485/K type thermocouple
Optical resolution	15:1、20:1
Protection level	IP65
Ambient temperature	0 ~ 60°C
Storage temperature	-20 ~ 80°C
Relative humidity	10–95%(non-condensing)
Material	stainless steel
Spectral range	8 ~ 14 μm
Response time	300 ms (95%)
Temperature measurement accuracy	±2%or±2℃ of the measured value
Repeatability	±1%or±1.5℃of the measured value
Dimension	113mm×ф18mm(length*diameter) 86mm×ф18mm(length*diameter) 46mm×ф18mm(length*diameter)
Cable length	1.5 m (standard), other special specifications (customized)
Emissivity	0.95 (can be adjusted by software)
Temperature range	0~200°C/0~300°C/0~400°C/0~500°C/0~600°C/0~800°C

	/0~1200°C/-20~100°C/-20~200°C/-20~300°C/-20~500°C /-20~600°C/-20~1200°C

1.2 Application

Infrared temperature sensors are widely used in food, power switch cabinets, metal materials, ceramics, graphite, printing and dyeing, microwaves, plastics, cables, rubber, chemicals, glass, composite materials, semiconductors, papermaking, textiles, metallurgy, refractory materials, pharmaceuticals, heating On-line measurement of furnace and other equipment temperature.

2. Working Principle

2.1 Infrared Temperature Measurement Principle

Any object radiates infrared energy outward, and the intensity of the radiation varies with temperature. Infrared thermometers generally use infrared radiation energy with a wavelength in the range of 0.8 μm to 18 μm. Infrared temperature sensor is a kind of optoelectronic sensor, which receives infrared radiation and converts it into electrical signal, and displays or outputs temperature through electronic circuit amplifier, linearization, and signal processing.

2.2 The Maximum Distance and Size of the Measured Point

The size of the measured object and the optical properties of the infrared thermometer determine the maximum distance between the measured object and the measuring head. In order to avoid measurement errors, the target to be measured should fill the field of view of the probe as much as possible. Therefore, the measured point should always be kept smaller than the measured object or at least the same size as the measured object.

2.3 Ambient Temperature

IR62 infrared temperature sensor can work in the range of ambient temperature 0-60℃. Otherwise, opt for a cooling jacket.

2.4 Lens Cleaning

The lens must be kept clean to avoid measurement errors or even damage to the lens due to dust, smoke and other pollutants. If there is dust on the lens, use lens cleaning paper dipped in absolute alcohol to wipe it.

2.5 Electromagnetic Interference

Please try to keep the infrared temperature sensor away from electromagnetic field sources (such as electric motors, motors, high-power cables, etc.) during installation, and add metal sleeves if necessary.

3. Installation

3.1 Mechanical Installation

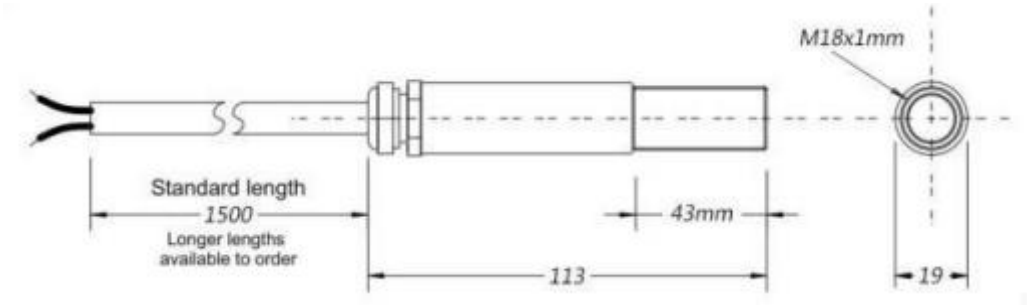
IR62 metal housing with M18×1 thread can be used for direct installation or by using a mounting bracket. The adjustable mounting bracket can make the adjustment of the measuring head more convenient. When adjusting the target to be measured and the measuring head, it must be ensured that the optical path is not blocked.

3.2 Wiring

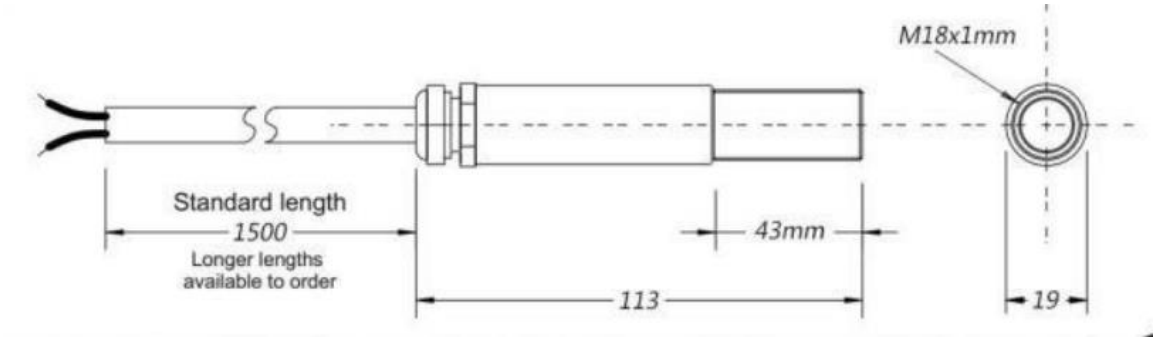
Type	Wire Color	Function
Current output (4～20mA)	Red	DC power +6~32V
	Black	Signal output 4~20mA+
	Transparent	Shielded wire
Voltage output (0~5V /0~10V/ K-type thermocouple)	Red	DC power＋ 6~28V/(11V~28V)
	White	DC power
	Blue	Signal output
	Transparent	Shielded wire
Data communication (Serial output RS485)	Red	DC power＋6~32V
	Black	DC power
	Green	D+/A
	Yellow	D-/B
	Transparent	Shielded wire

4. Dimensions

113mm×ф18mm (length*diameter)



46mm×ф18mm (length*diameter)



5. Communication Protocol

5.1 Serial Parameter

Baud rate	Data bit	Check Digit	Stop bit
9600	8	None	1

5.2 Register Description

Register name	**Register address**	**Data length**	**Attributes**	**Data range (decimal)**	**Note**
Device address	0x0001	2 bytes	read/write	1-255	
Baud rate	0x0002	2 bytes	read/write	0~3	0:2400、1:4800、2:9600、3:19200
Emissivity	0x0003	2 bytes	read/write	0.10~1.10	Example: 0.95 emissivity expressed as decimal 95
Infrared temperature value	0x0004	2 bytes	Read only	-50.0~600.0	Determined according to the upper and lower limits of the actual product range

5.3 Read register command: 0x03

Read Register Data Structure

Device address	Order	First register address	Number of registers	CRC check
1Byte	1Byte	2Byte	2Byte	2Byte

Infrared temperature measurement module returns data structure

Device address	Order	Data length	Data content	CRC check
1Byte	1Byte	1Byte	...	2Byte

Example: read baud rate and emissivity

Read registers (0X0002~0X0003, baud rate and emissivity):

Device address	Order	First register address	Number of registers	CRC check
0x01	0x03	0x0002	0x0002	0x65CB

Infrared temperature measurement module response data:

Device address	Order	Data length	Data content	CRC check
0x01	0x03	0x04	0x0002 0x005F	0x1BCB
			0x0002: baud rate 9600bps 0x005F: emissivity 0.95	

Example: read temperature

Read the register (0X0004, the infrared temperature value):

Device address	Order	First register address	Number of registers	CRC check
0x01	0x03	0x0004	0x0001	0xC5CB

Infrared temperature measurement module response data:

Device address	Order	Data length	Data content	CRC check
0x01	0x03	0x02	0x01F4	0xB853
			0x01F4: infrared temperature =50.0℃	

5.4 Modify register command：0x10

Modify register data structure：

Device address	Order	First register address	Number of registers	Data length	Data content	CRC check
1Byte	1Byte	2Byte	2Byte	N Byte	...	2Byte

Infrared temperature measurement module returns data structure：

Device address	Order	First register address	Number of registers	CRC check
1Byte	1Byte	2Byte	2Byte	2Byte

Example

Modify device address, baud rate and emissivity:

Modify registers (0X0001~0X0003, baud rate and emissivity):

Device address	Order	First register address	Number of registers	Data length	Data content	CRC check
0x01	0x10	0x0001	0x0003	0x06	0x0002 0x0003 0x0064	0x3F6E
					0x0002:device address 2 0x0003:baud rate 19200bps 0x0064:emissivity 1.00	

Infrared temperature measurement module response data:

Device address	Order	First register address	Number of registers	CRC check
0x01	0x10	0x0001	0x0003	0xD1C8

When the device address is 0, the modification command 0x10 is broadcast status, and all infrared temperature measurement modules on the bus will receive the data and update the registers according to the received content. In the broadcast state, the infrared module will not return data.

5.5 CRC16 checksum generating function

```
/*****************************************************************************
* Function Name     : crc16
* Input             : data buffer pointer：puchMsg ，  data length：usDataLen
* Return            : 16-bit CRC checksum
* Description       : generate 16-bit CRC check code
*****************************************************************************/ INT16U
crc16(INT8U *puchMsg, INT8U usDataLen)
{
    INT8U uchCRCHi=0xFF ; /* High CRC byte initialization */ INT8U uchCRCLo=0xFF ; /*
    Low CRC byte initialization */
    INT16U uIndex;          /* Index in CRC cycle */

    while(usDataLen--)      /* transmit message buffer    */
    { uIndex =uchCRCHi^*puchMsg++ ; /* calculate CRC */
      uchCRCHi=uchCRCLo^auchCRCHi[uIndex] ; uchCRCLo=auchCRCLo[uIndex];
    }
    return (uchCRCHi<<8|uchCRCLo);
}
    /* CRC High byte value table */ const INT8U code auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41,  0x00, 0xC1,  0x81, 0x40, 0x01,  0xC0,  0x80,  0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41,  0x01, 0xC0,  0x80, 0x41, 0x00,  0xC1,  0x81,  0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40,  0x01, 0xC0,  0x80, 0x41, 0x01,  0xC0,  0x80,  0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40,  0x00, 0xC1,  0x81, 0x40, 0x01,  0xC0,  0x80,  0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41,  0x00, 0xC1,  0x81, 0x40, 0x00,  0xC1,  0x81,  0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40,  0x01, 0xC0,  0x80, 0x41, 0x00,  0xC1,  0x81,  0x40,
```

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40

} ;

/* CRC Low byte value table */ const INT8U code auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07,
0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F,
0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08,
0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E,
0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5,
0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11,
0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD,
0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39,
0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A,
0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4,
0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,

```
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
}
```