



Temperature Range, Field-of-View

CT - □N - 485

Temp. code	Measurement range	FOV (field of view)
200	-20 ... 400°C	7.16°
300	-20 ... 500°C	3.814°
1000	-20 ... 1000°C	2.886°

e.g. Model CT-200N-485 has a 7.16° field of view and provides object temperatures of -20...400°C.

Product Specifications

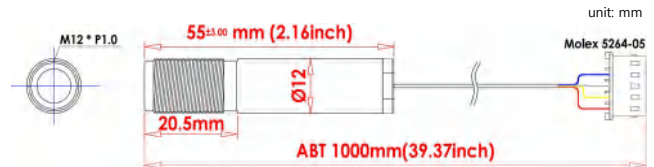
If not otherwise noted, 25°C ambient temperature, 5V supply voltage were applied.

Parameter	min	Typ	Max	Unit
Supply voltage	4.75	5	12	V
Supply current		15		mA
Spectral range	8	-	14	μm
Operating temperature	-20		70	°C
IR refresh rate		10	10	Hz
Accuracy(*)		±2		%
Resolution digital		0.1		°C
Emission coefficient	0.1	0.97	1.0	ε
Standard start-up time		1	2	sec
Stabilization time	1			min
Dimensions	Ø12 x 55mm(long)			
Thread mounting	M12 x 1mm pitch			
Cable length	about 1m (39.37 inch)			
Weight with cable	36g			
Cable interface	molex 5264-05			
Communication interface/ protocol	RS-485/ Modbus-RTU			
Relative humidity	95% Max. non-condensing			

*: ±2% of reading or ±2°C whichever is greater.

Accuracy is only effective if the object is fully covered by the sensor's FOV and applicable to stable temperature conditions.

Dimensions / Pins and Wiring colors

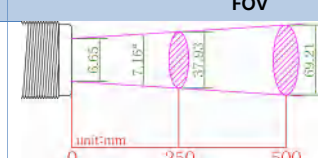
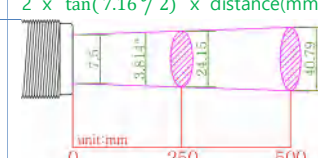
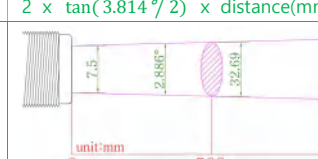


The shield wire is connected to the GND wire.

No.	Wire Color	Description
1	Red	VDD (5V)
2	Yellow or Black	Ground
3	White	RS485 D-
4	Blue or Green	RS485 D+
5	None	None

Calculate Field of View

The FOV determines the size of the infrared measurement area according to the distance.

Angle	FOV
7.16°	 $2 \times \tan(7.16^\circ / 2) \times \text{distance}(\text{mm}) + 6.65 (\text{mm})$
3.814°	 $2 \times \tan(3.814^\circ / 2) \times \text{distance}(\text{mm}) + 7.5 (\text{mm})$
2.886°	 $2 \times \tan(2.886^\circ / 2) \times \text{distance}(\text{mm}) + 7.5 (\text{mm})$

Accessories

	nut (assembled to the body.)	2pcs
	Protective cap (Remove when using)	1pc
	Molex 5267-05A-X	1pc

Modbus-RTU Register Table

- BaudRate: 19,200 bps(fixed), data bit: 8, stop bit: 1, parity: none, flow control: none.

- R = Read - W = Write (single write)

Address		Length (short)	Description	R/W
Dec	Hex			
40,000	0x9C40	1	Device ID (1 ~ 200), Modbus broadcast not supported.	R/W
40,001	0x9C41	1	Emissivity (10~100. default : 97) (*)	R/W
40,002	0x9C42	1	Object temperature	R
40,003	0x9C43	1	Ambient temperature	R
40,004	0x9C44	1	Average Filter (1~10, default : 10) (**)	R/W

※: "97" means emissivity "0.97". To adjust the emissivity to 0.95, write 95 not 0.95.

※*: number of average filter array elements. Affected by noise reduction of Object temperature and peak temperature measurement time.
Time to peak temperature: up to 1 sec (default: 10)

Support Modbus function codes

- Read Holding Registers 03 (0x03)
- Write Single Register: 06 (0x06)

Object Temperature: To, Ambient Temperature: Ta

To is the object temperature derived from thermopile and ambient sensor outputs.
Ta stands for ambient temperature.

0x016D(read data) = 365(dec) → means 36.5°C

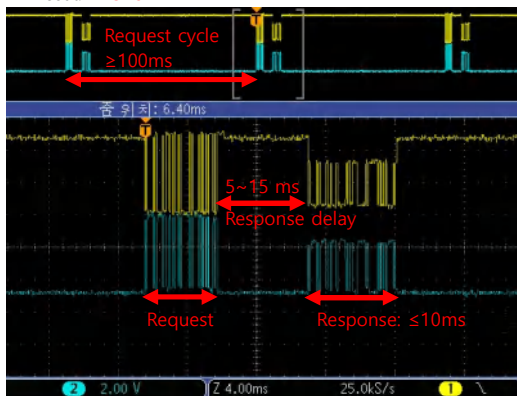
0xFFFF1(read data) → 0x000F(two's complement)=15 → means -1.5°C

※ Output Data Limit

Model	Temperature range
CT-200N-485	-20 ... 450°C
CT-300N-485	-20 ... 550°C
CT-1000N-485	-20 ... 1100°C

Request & Response timing

- Request cycle: ≥100ms
- First data request time after Power-on: ≥ 1 sec
- Timeout: ≥ 25ms



Note. If there is an error in the request sequence (including crc), there is no response data.

Products handling precaution

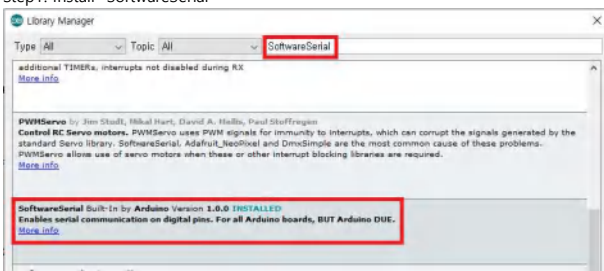
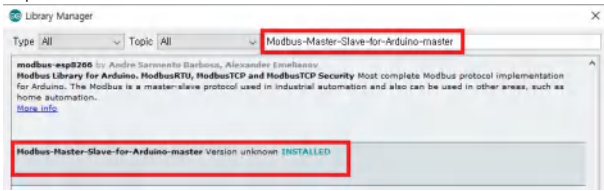
- ※ When it comes to dust removal by air, the best method is to use a blower, and to avoid using compressed air.
- ※ Do not press the lens with your hands or any other object.
- ※ Do not scratch the lens surface with sharp objects.
- ※ Voluntary disassembly and modification of the product is prohibited.
- ※ Avoid direct sunlight, chemical substance, heat or fire.
- ※ Water resistance is not guaranteed.

CT-N Series Specification

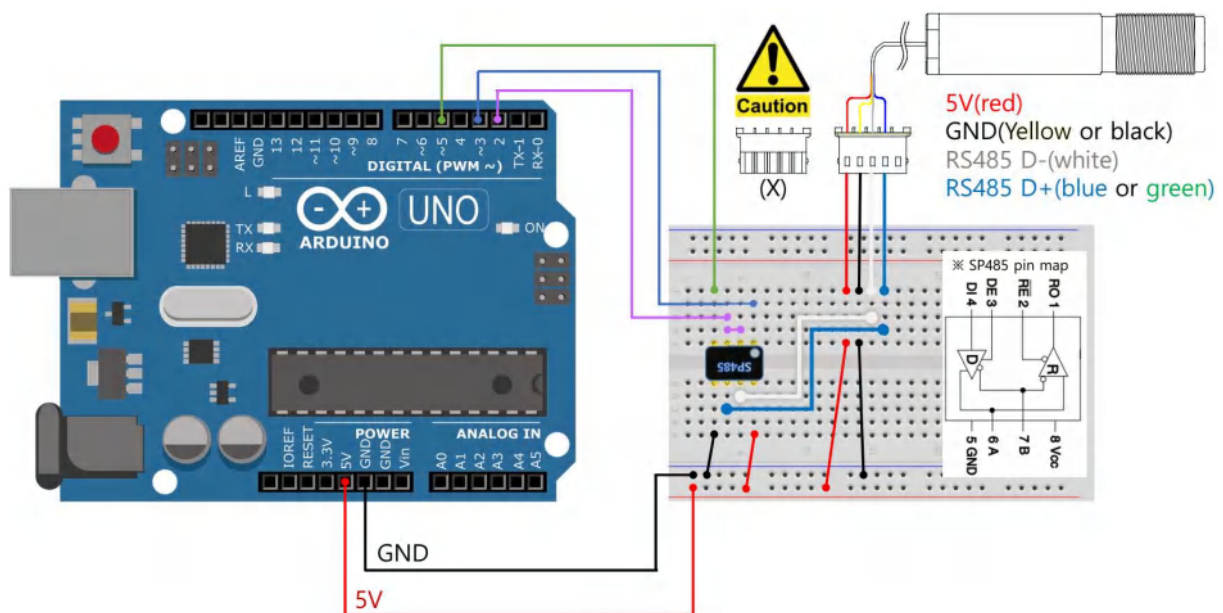
Compact Non-contact Infrared Thermometer

Tutorial (Arduino UNO)

- Requirements

Hardware	Software
Arduino Uno x 1 Half-Duplex RS-485 Transceivers IC (SP485) x 1 CT-N Series Infrared thermometer x 1	Arduino IDE. Example code download: https://diwellshop.cafe24.com/web/en/CT-N/CT-N_Arduino.zip Download and install the [MsTimer2], [SoftwareSerial] and [Modbus-Master-Slave-for-Arduino-master] in Library Manager. Step1. Install "SoftwareSerial"
Note. This example is a Modbus 485 example using software serial, not hardware serial. Therefore, be sure to refer to the <i>Connection Diagram PDF</i> for circuit connection.	
Step2 If the Modbus library search fails, download the zip file from the link below and then add the library manually. https://diwellshop.cafe24.com/web/en/lib/Modbus-Master-Slave-for-Arduino-master.zip	Step2. Install "Modbus-Master-Slave-for-Arduino-master" 
	Step3. Install "MsTimer2" 

- Connection Diagram



- Sample Code

```
#include <MsTimer2.h>           // Timer library.
#include <ModbusRtu.h>          // Modbus library.
#include "SoftwareSerial.h"     // SoftwareSerial library
#define DISABLE 0
#define ENABLE 1
#define DE_RE 2
#define USING_SOFTWARESERIAL 4
#define ID 1

int8_t Timer_Flag = 0, Data_Print = DISABLE;
uint16_t au16data[2];
int16_t Object, Ambient;

SoftwareSerial mySerial(3, 5); // RX 3, TX 5
Modbus master(0,USING_SOFTWARESERIAL,DE_RE); // Modbus Master, 4:Using SoftwareSerial , DE/RE 2
modbus_t telegram;           // Master query structure

void setup() {
    Serial.begin(9600); // for Serial Monitor (Ctrl + Shift + M )

    master.begin(&mySerial, 19200); // begin the ModBus object.
    master.setTimeout( 25 ); // Modbus timeout : 25 ms

    Serial.println("Waiting for sensor initialization time");
    delay(1000); // Wait for sensor initialization time

    MsTimer2::set(500, timerISR); // Timer interval : 500ms.
    MsTimer2::start(); // Timer Start
}

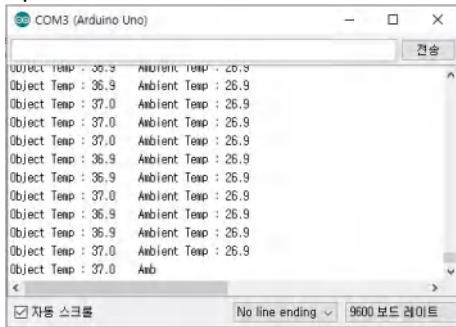
void loop() {
    if(Timer_Flag) { // Check timer interrupt
        Timer_Flag = 0;
        Transfer_Data(ID); // Request data transmission : Timer cycle
    }
    if(master.getState() == COM_WAITING) { // Get modbus master state : waiting for answer
        master.poll(0);
        Data_Print = ENABLE;
    }
    if ((master.getState() == COM_IDLE) && (Data_Print == ENABLE)) { // Get modbus master state : idle
        if(master.getLastError() == 0){ // Get the last error in the protocol processor (0: No error)
            Object = au16data[0];
            Ambient = au16data[1];

            Serial.print("Object Temp: ");
            Serial.print(float(Object)/10, 1); // celsius
            Serial.print("    Ambient Temp: ");
            Serial.println(float(Ambient)/10, 1);
        }
        else if(master.getLastError() == NO_REPLY) { // Time-out
            Serial.println("No reply.");
        }
        Data_Print = DISABLE;
    }
}

void timerISR() { Timer_Flag = 1; } // Timer Interrupt Service Routine

void Transfer_Data(uint8_t uid) {
    if( (uid == 0) &&(uid>200) ) { // ID : 1~200
        uid = 1;
    }
    telegram.u8id = uid; // Do not change the parameter values below.
    telegram.u8fct = 3; // slave ID
    telegram.u16RegAdd = 40002; // function code = 03 (Read Holding Registers)
    telegram.u16CoilsNo = 2; // start address in slave
    telegram.au16reg = au16data; // number of elements to read
    master.query( telegram ); // pointer to a memory array in the Arduino
    // send query
}
```

- Expected Results.



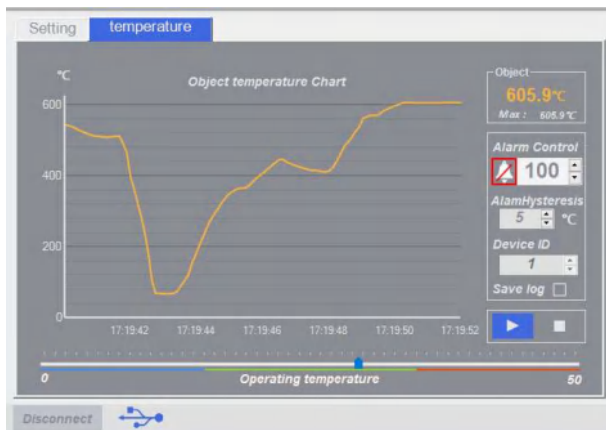
PC Software

The program runs in the Windows 10 environment.

It is not guaranteed to be used on other OS.

For more information, refer to the Test Board manual.

https://www.diwellshop.com/web/en/CT-N/CT-N_Testboard_en.pdf



Additional information

Manufacturer: DIWELL Electronics Co., Ltd. (South Korea)

Technical support: <mailto:expoeb2@diwell.com>, <mailto:dsjeong@diwell.com>

Revision history

Version	Date(Y,M,D)	Description
1.0.0	2022. 5. 9.	First version is released
1.0.1	2022. 7. 14.	Added Modbus library download link.