

SKU:SEN0358 (<https://www.dfrobot.com/product-2173.html>)



([https://www.dfrobot.com/product-](https://www.dfrobot.com/product-2173.html)

2173.html)

Introduction

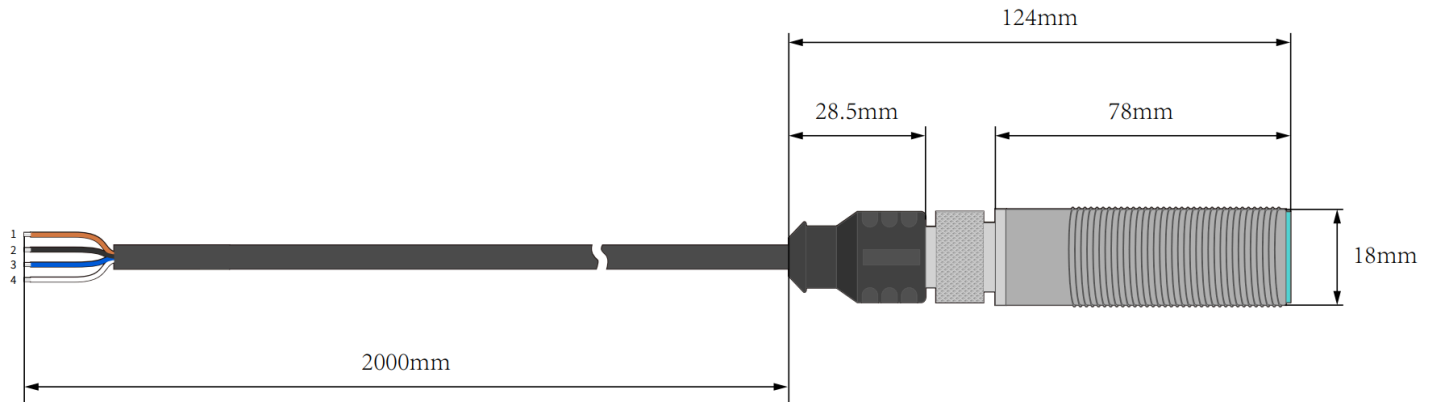
This is 200KHz high-frequency ultrasonic ranging sensor with IP65 protection grade. The sensor uses the RS485 interface that follows the standard Modbus-RTU communication protocol, featuring reliable communication. The sensor's slave address and serial port parameter can be revised according to the actual use, so it can be conveniently used with all kinds of industrial controlling machines. In addition, the sensor comes with flexible temperature compensation function that allows users to use the sensor built-in temperature compensation or external temperature compensation according to their needs, which makes it applicable to various complicated application scenarios.

Besides that, the URM14 adopts high-frequency and stable ultrasonic transducer that offers 100mm~1500mm effective measuring range for flat reflecting wall, 1mm accuracy, and $\pm 1\%$ error. Compared with the general ultrasonic ranging sensor working at 40 KHZ acoustic frequency, this ranging sensor has higher measurement accuracy and shorter measurement period, which makes it very suitable for high precision and fast response ranging applications. In addition, the sensor has built-in noise level evaluation function, which can realize adaptive ranging in complex scenes. Users can also obtain noise level to optimize the working conditions of the sensor, so as to obtain the highest precision measurement data.

Specification

- Operating Voltage: DC 7-15V
- Max Instantaneous Current: 350mA
- Effective Measuring Range: 100-1500mm
- Distance Resolution: 0.1mm
- Measuring Accuracy: 1mm
- Distance Error: $\pm 0.1\%$
- Temperature Resolution: 0.1°C
- Temperature Error: $\pm 1^\circ\text{C}$
- Measuring Frequency: 30HZ
- Operating Temperature: $-20^\circ\text{C} \sim + 80^\circ\text{C}$
- Sensor Acoustic Frequency: 200KHz $\pm 4\%$
- Directional Angle: $12^\circ \pm 2^\circ (-6\text{dB})$
- Protection Grade: IP65
- Communication interface: RS485

Board Overview



Module Wire Connecting Order:

- Orange-----VCC
- Black-----GND
- Blue-----RS485-B
- White-----RS485-A

Register

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x00	1	Module PID Register	R	0x0000-0xFFFF	0x0002	Pi r
0x01	1	Module VID Register	R	0x0000-0xFFFF	0x0010	Vi re

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x02	1	Module Address Register	R/W	0x0001-0x00F7	0x000C	V a c u u t i o n S i g n a l r e
0x03	1	Serial parameter control register 1	R/W	0x0000-0xFFFF	0x0005	M o d e s t r i m e n t S i g n a l r e

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x04	1	Serial parameter control register 2	R/W	0x0000-0xFFFF	0x0001	M H 0: N -C 0: -1 0: O -1 o n -2 O S o re
0x05	1	Distance register	R	0x0000-0xFFFF	0xFFFF	TI m m 0.
0x06	1	Onboard temperature data register	R	0x0000-0xFFFF	0x0000	TI L o --

data register

SE

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x07	1	External temperature compensation data register	R/W	0x0000-0xFFFF	0x0000	V te th te cc re ui

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x08	1	Control register	R/W	0x0000-0xFFFF	0x0004	bi 0- te cc 1- te cc fu w tc cc re bi 0- cc 1- cc bi 0- 1- bi In m bi di di re re

Register Address	Number	Name	Read/Write	Data Range	Default Value	D
0x09	1	Electrical noise level register	R	0x0000-0x0A	0x0000	

SEN0358 Register Read/Write Tutorial

Requirements

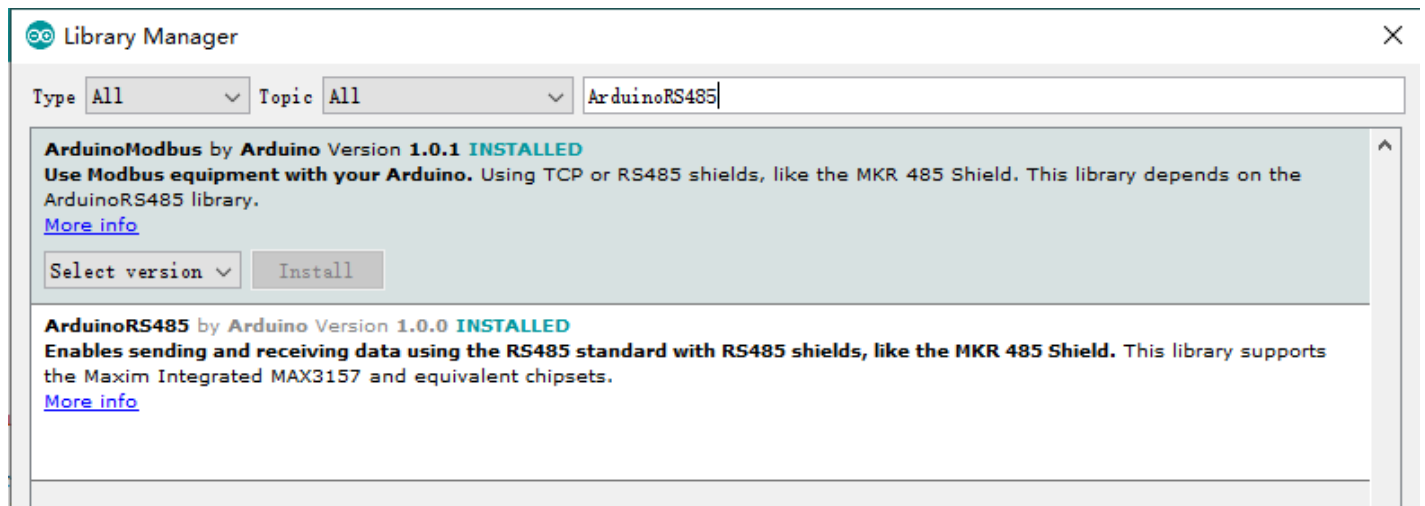
- Hardware
 - Arduino Leonardo (<https://www.dfrobot.com/product-832.html>) x 1 (RS485 to TTL needs to occupy one serial port, so we recommend you to use device with more than 2 serial ports. Since Arduino Modbus takes up a lot

device with more than 2 serial ports. Since Arduino Modbus takes up a lot of memory, it is suggested to use Arduino Mega2560 controller.)

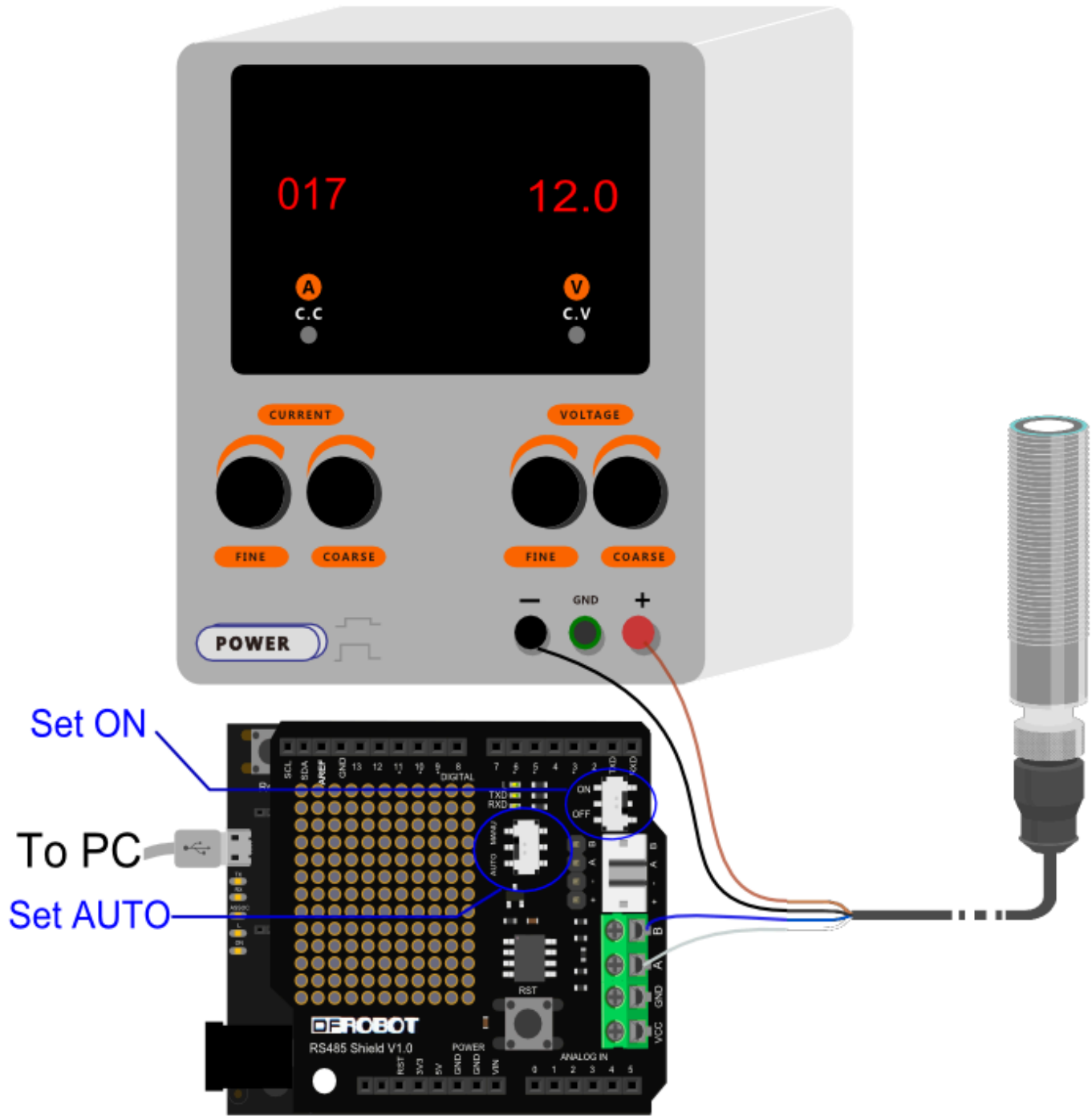
- RS485 Shield for Arduino
(https://wiki.dfrobot.com/Arduino_RS485_Shield_SKU_DFR0259) x1
- DC Power Supply x1
- USB Data Cable (Connect the Arduino board to a computer via the USB cable)

- **Software**

- Arduino IDE (<https://www.arduino.cc/en/Main/Software>)
- Open Library Manager(Ctrl+Shift+I) in Arduino IDE, find and install ArduinoModbus and ArduinoRS485 Libraries.



Connection Diagram



Read the detected distance

```

/*****
   This code tests the range finder function of the URM14 ultrasonic sensor
   @ author : roker.wang@dfrobot.com
   @ data   : 11.08.2020
   @ version: 1.0
 *****/
#include <ArduinoModbus.h>
#include <ArduinoRS485.h>

#define SLAVE_ADDR ((uint16_t)0x0C)

#define TEMP_CPT_SEL_BIT ((uint16_t)0x01)
#define TEMP_CPT_ENABLE_BIT ((uint16_t)0x01 << 1)
#define MEASURE_MODE_BIT ((uint16_t)0x01 << 2)
#define MEASURE_TRIG_BIT ((uint16_t)0x01 << 3)

typedef enum{
    ePid,
    eVid,
    eAddr,
    eComBaudrate,
    eComParityStop,
    eDistance,
    eInternalTempreture,
    eExternTempreture,
    eControl,
    eNoise
}eRegIndex_t;//Sensor register index

/*
 * @brief Read data from holding register of client
 *
 * @param addr : Address of Client

```

```

    *@param reg: Reg index
    *@return data if execute successfully, false 0xffff.
    */
uint16_t readData(uint16_t addr, eRegIndex_t reg)
{
    uint16_t data;
    if (!ModbusRTUClient.requestFrom(addr, HOLDING_REGISTERS, reg, 1)){
        Serial.print("failed to read registers! ");
        Serial.println(ModbusRTUClient.lastError());
        data = 0xffff;
    }else{
        data = ModbusRTUClient.read();
    }
    return data;
}

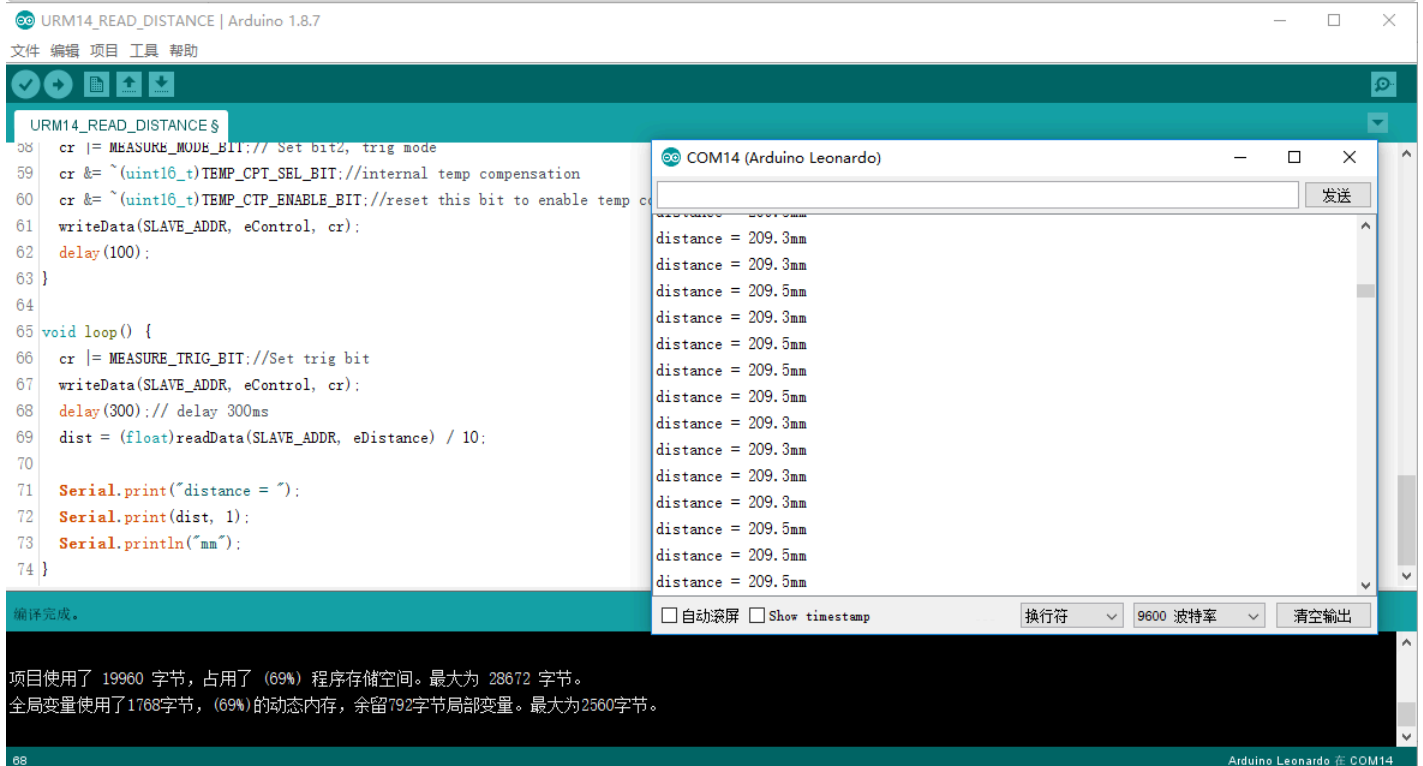
/*
    *@brief write data to holding register of client
    *
    *@param addr : Address of Client
    *@param reg: Reg index
    *@param data: The data to be written
    *@return 1 if execute successfully, false 0.
    */
uint16_t writeData(uint16_t addr, eRegIndex_t reg, uint16_t data)
{
    if (!ModbusRTUClient.holdingRegisterWrite(addr, reg, data)){
        Serial.print("Failed to write coil! ");
        Serial.println(ModbusRTUClient.lastError());
        return 0;
    }else
        return 1;
}

float dist;
volatile uint16_t cr = 0;
void setup() {
    ModbusRTUClient.begin(19200);
    Serial.begin(9600);
    cr |= MEASURE_MODE_BIT;//Set bit2 , Set to trigger mode
    cr &= ~(uint16_t)TEMP_CPT_SEL_BIT;//Select internal temperature compensation
    cr &= ~(uint16_t)TEMP_CPT_ENABLE_BIT;//enable temperature compensation

```

```
writeData(SLAVE_ADDR, eControl, cr); //Writes the setting value to the control register
delay(100);
}

void loop() {
  cr |= MEASURE_TRIG_BIT; //Set trig bit
  writeData(SLAVE_ADDR, eControl, cr); //Write the value to the control register
  delay(300); //Delay of 300ms (minimum delay should be greater than 30ms) is to allow the sensor to stabilize
  dist = (float)readData(SLAVE_ADDR, eDistance) / 10; //Read distance register, divide by 10 to get cm
  Serial.print("distance = ");
  Serial.print(dist, 1);
  Serial.println("mm");
}
```



The screenshot shows the Arduino IDE interface. The main editor window displays the code from the previous block, with line numbers 58 to 74. The serial monitor window, titled 'COM14 (Arduino Leonardo)', shows the output of the program, displaying distance measurements in millimeters (mm) such as 'distance = 209.3mm', 'distance = 209.5mm', etc. The status bar at the bottom indicates '编译完成。' (Compilation completed) and provides memory usage statistics: '项目使用了 19960 字节, 占用了 (69%) 程序存储空间。最大为 28672 字节。全局变量使用了 1768 字节, (69%) 的动态内存, 余留 792 字节局部变量。最大为 2560 字节。' (The project used 19960 bytes, occupying 69% of program memory space. Maximum is 28672 bytes. Global variables used 1768 bytes, 69% of dynamic memory, leaving 792 bytes for local variables. Maximum is 2560 bytes.)

Read Onboard Temperature

```

/*****
   This code tests the temperature measurement function of the URM14 ultrasonic sensor
   @ author : roker.wang@dfrobot.com
   @ data   : 11.08.2020
   @ version: 1.0
   RX(TTL-RS485转接板) -> TX1/D1 (Arduino Leonardo) TX (TTL-RS485转接板) -> TX2/D2
   *****/
#include <ArduinoModbus.h>
#include <ArduinoRS485.h>

#define SLAVE_ADDR ((uint16_t)0x0C)

#define TEMP_CPT_SEL_BIT ((uint16_t)0x01)
#define TEMP_CPT_ENABLE_BIT ((uint16_t)0x01 << 1)
#define MEASURE_MODE_BIT ((uint16_t)0x01 << 2)
#define MEASURE_TRIG_BIT ((uint16_t)0x01 << 3)

typedef enum{
    ePid,
    eVid,
    eAddr,
    eComBaudrate,
    eComParityStop,
    eDistance,
    eInternalTemperature,
    eExternalTemperature,
    eControl,
    eNoise
}eRegIndex_t;//Sensor register index

/*
 * @brief Read data from holding register of client
 *

```

```

    *@param addr : Address of Client
    *@param reg: Reg index
    *@return data if execute successfully, false 0xffff.
    */
uint16_t readData(uint16_t addr, eRegIndex_t reg)
{
    uint16_t data;
    if (!ModbusRTUClient.requestFrom(addr, HOLDING_REGISTERS, reg, 1)){
        Serial.print("failed to read registers! ");
        Serial.println(ModbusRTUClient.lastError());
        data = 0xffff;
    }else{
        data = ModbusRTUClient.read();
    }
    return data;
}

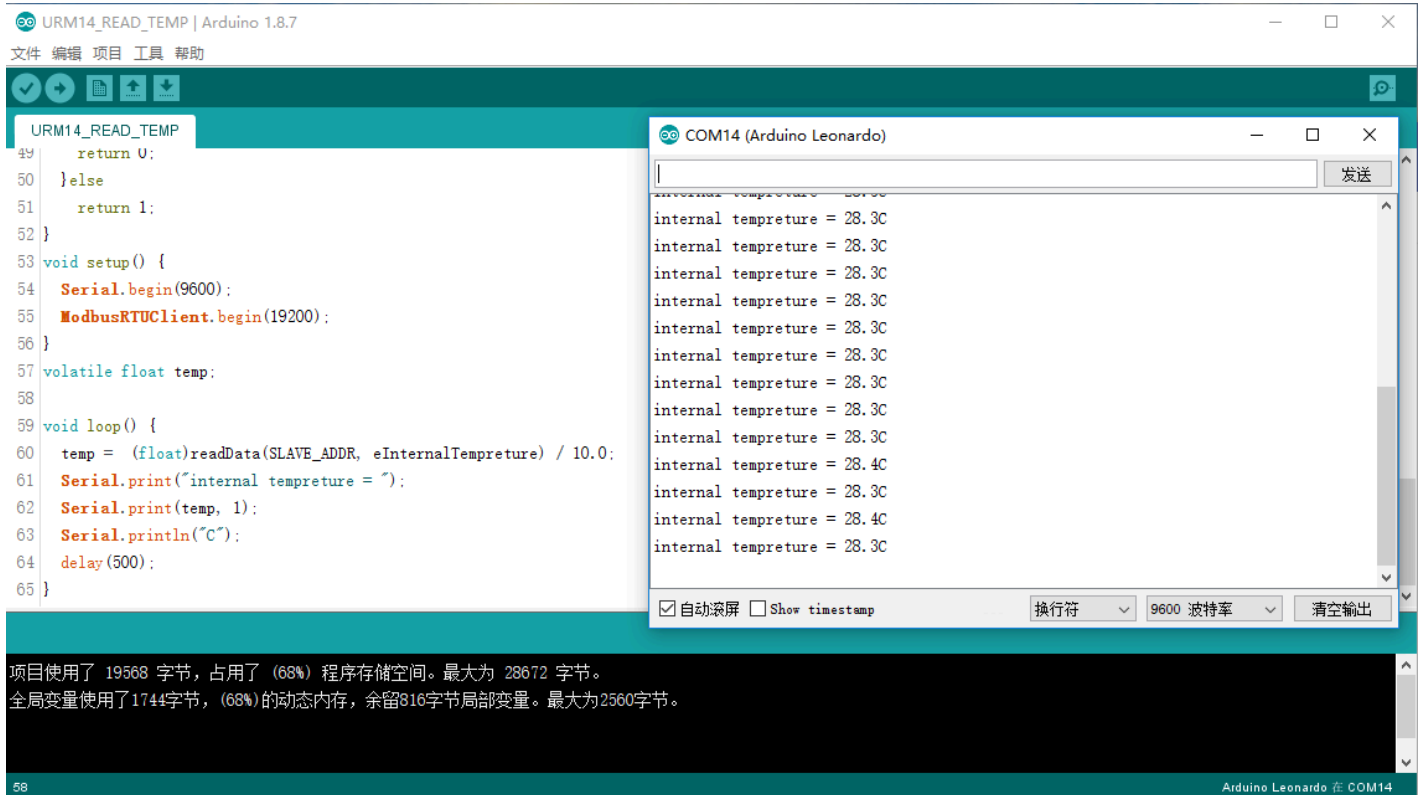
/*
    *@brief write data to holding register of client
    *
    *@param addr : Address of Client
    *@param reg: Reg index
    *@param data: The data to be written
    *@return 1 if execute successfully, false 0.
    */
uint16_t writeData(uint16_t addr, eRegIndex_t reg, uint16_t data)
{
    if (!ModbusRTUClient.holdingRegisterWrite(addr, reg, data)){
        Serial.print("Failed to write coil! ");
        Serial.println(ModbusRTUClient.lastError());
        return 0;
    }else
        return 1;
}

void setup() {
    Serial.begin(9600);
    ModbusRTUClient.begin(19200);
}

volatile float temp;
void loop() {
```

```
temp = (float)readData(SLAVE_ADDR, eInternalTempreture) / 10.0; //Read the t
Serial.print("internal tempreture = ");
Serial.print(temp, 1);
Serial.println("C");

delay(500);
}
```



The screenshot shows the Arduino IDE interface. The left pane displays the code for 'URM14_READ_TEMP'. The right pane shows the serial monitor output for 'COM14 (Arduino Leonardo)'. The serial monitor displays a series of 'internal tempreture = 28.3C' and 'internal tempreture = 28.4C' messages. The status bar at the bottom indicates that the project used 19568 bytes of program memory (68% of 28672 bytes) and 1744 bytes of dynamic memory (68% of 2560 bytes).

```
URM14_READ_TEMP | Arduino 1.8.7
文件 编辑 项目 工具 帮助
URM14_READ_TEMP
49   return U;
50 }else
51   return 1;
52 }
53 void setup() {
54   Serial.begin(9600);
55   ModbusRTUClient.begin(19200);
56 }
57 volatile float temp;
58
59 void loop() {
60   temp = (float)readData(SLAVE_ADDR, eInternalTempreture) / 10.0;
61   Serial.print("internal tempreture = ");
62   Serial.print(temp, 1);
63   Serial.println("C");
64   delay(500);
65 }

COM14 (Arduino Leonardo)
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.3C
internal tempreture = 28.4C
internal tempreture = 28.3C
internal tempreture = 28.4C
internal tempreture = 28.3C

[自动滚屏] [Show timestamp] 换行符 9600 波特率 清空输出

项目使用了 19568 字节, 占用了 (68%) 程序存储空间。最大为 28672 字节。
全局变量使用了1744字节, (68%) 的动态内存, 余留816字节局部变量。最大为2560字节。

58 Arduino Leonardo 在 COM14
```


Revise Module Address

```

/*****
    This code tests the address modification function of the URM14 ultrasonic
    @ author : roker.wang@dfrobot.com
    @ data   : 11.08.2020
    @ version: 1.0
    RX(TTL-RS485转接板) -> TX1/D1 (Arduino Leonardo) TX (TTL-RS485转接板) -
*****/
#include <ArduinoModbus.h>
#include <ArduinoRS485.h>

#define PUBLIC_ADDR ((uint16_t)0x00)
#define SLAVE_ADDR  ((uint16_t)0x0C)

#define TEMP_CPT_SEL_BIT ((uint16_t)0x01)
#define TEMP_CPT_ENABLE_BIT ((uint16_t)0x01 << 1)
#define MEASURE_MODE_BIT ((uint16_t)0x01 << 2)
#define MEASURE_TRIG_BIT ((uint16_t)0x01 << 3)

typedef enum{
    ePid,
    eVid,
    eAddr,
    eComBaudrate,
    eComParityStop,
    eDistance,
    eInternalTempreture,
    eExternTempreture,
    eControl,
    eNoise
}eRegIndex_t;//Sensor register index

/*
 *@brief Read data from holding register of client
 *

```

```

  *@param addr : Address of Client
  *@param reg: Reg index
  *@return data if execute successfully, false 0xffff.
  */
uint16_t readData(uint16_t addr, eRegIndex_t reg)
{
  uint16_t data;
  if (!ModbusRTUClient.requestFrom(addr, HOLDING_REGISTERS, reg, 1)){
    Serial.print("failed to read registers! ");
    Serial.println(ModbusRTUClient.lastError());
    data = 0xffff;
  }else{
    data = ModbusRTUClient.read();
  }
  return data;
}

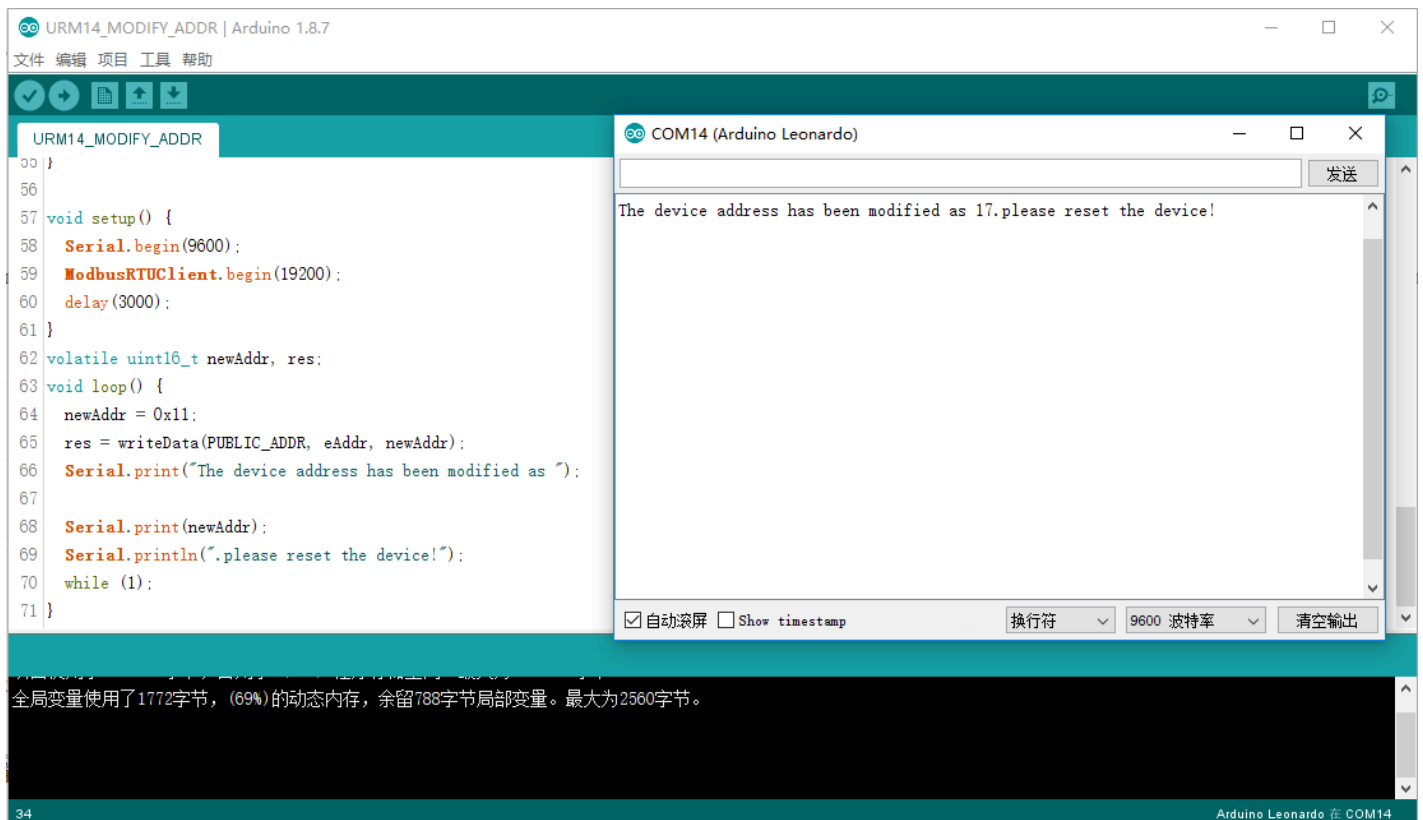
/*
  *@brief write data to holding register of client
  *
  *@param addr : Address of Client
  *@param reg: Reg index
  *@param data: The data to be written
  *@return 1 if execute successfully, false 0.
  */
uint16_t writeData(uint16_t addr, eRegIndex_t reg, uint16_t data)
{
  if (!ModbusRTUClient.holdingRegisterWrite(addr, reg, data)){
    Serial.print("Failed to write coil! ");
    Serial.println(ModbusRTUClient.lastError());
    return 0;
  }else
    return 1;
}

void setup() {
  Serial.begin(9600);
  ModbusRTUClient.begin(19200);
  delay(3000);
}
volatile uint16_t newAddr, res;
void loop() {

```

```
newAddr = 0x11;
res = writeData(PUBLIC_ADDR, eAddr, newAddr);//Writes the new address value
Serial.print("The device address has been modified as ");

Serial.print(newAddr);
Serial.println(".please reset the device!");
while (1);
}
```



The screenshot displays the Arduino IDE interface. The main window shows the sketch 'URM14_MODIFY_ADDR' with the following code:

```
50 }
56
57 void setup() {
58   Serial.begin(9600);
59   ModbusRTUClient.begin(19200);
60   delay(3000);
61 }
62 volatile uint16_t newAddr, res;
63 void loop() {
64   newAddr = 0x11;
65   res = writeData(PUBLIC_ADDR, eAddr, newAddr);
66   Serial.print("The device address has been modified as ");
67
68   Serial.print(newAddr);
69   Serial.println(".please reset the device!");
70   while (1);
71 }
```

The serial monitor window, titled 'COM14 (Arduino Leonardo)', shows the output: 'The device address has been modified as 17.please reset the device!'. The monitor settings are set to 9600 baud rate and 13N1 parity. The status bar at the bottom indicates 'Arduino Leonardo 在 COM14'.

全局变量使用了1772字节, (69%)的动态内存, 余留788字节局部变量。最大为2560字节。

Revise Module Baud Rate

```

/*****
   This code tests the baudrate modification function of the URM14 ultrasonic sensor
   @ author : roker.wang@dfrobot.com
   @ data   : 11.08.2020
   @ version: 1.0
   RX(TTL-RS485转接板) -> TX1/D1 (Arduino Leonardo) TX (TTL-RS485转接板) -> TX2/D2
   *****/
#include <ArduinoModbus.h>
#include <ArduinoRS485.h>

#define BAUDRATE_DEFAULT ((uint32_t)19200)
#define SLAVE_ADDR ((uint16_t)0x0C)

#define TEMP_CPT_SEL_BIT ((uint16_t)0x01)
#define TEMP_CPT_ENABLE_BIT ((uint16_t)0x01 << 1)
#define MEASURE_MODE_BIT ((uint16_t)0x01 << 2)
#define MEASURE_TRIG_BIT ((uint16_t)0x01 << 3)

typedef enum{
    ePid,
    eVid,
    eAddr,
    eComBaudrate,
    eComParityStop,
    eDistance,
    eInternalTempreture,
    eExternTempreture,
    eControl,
    eNoise
}eRegIndex_t;//Sensor register index

/*
 * @brief Read data from holding register of client
 *

```

```

    *@param addr : Address of Client
    *@param reg: Reg index
    *@return data if execute successfully, false 0xffff.
    */
uint16_t readData(uint16_t addr, eRegIndex_t reg)
{
    uint16_t data;
    if (!ModbusRTUClient.requestFrom(addr, HOLDING_REGISTERS, reg, 1)){
        Serial.print("failed to read registers! ");
        Serial.println(ModbusRTUClient.lastError());
        data = 0xffff;
    }else{
        data = ModbusRTUClient.read();
    }
    return data;
}

/*
    *@brief write data to holding register of client
    *
    *@param addr : Address of Client
    *@param reg: Reg index
    *@param data: The data to be written
    *@return 1 if execute successfully, false 0.
    */
uint16_t writeData(uint16_t addr, eRegIndex_t reg, uint16_t data)
{
    if (!ModbusRTUClient.holdingRegisterWrite(addr, reg, data)){
        Serial.print("Failed to write coil! ");
        Serial.println(ModbusRTUClient.lastError());
        return 0;
    }else
        return 1;
}

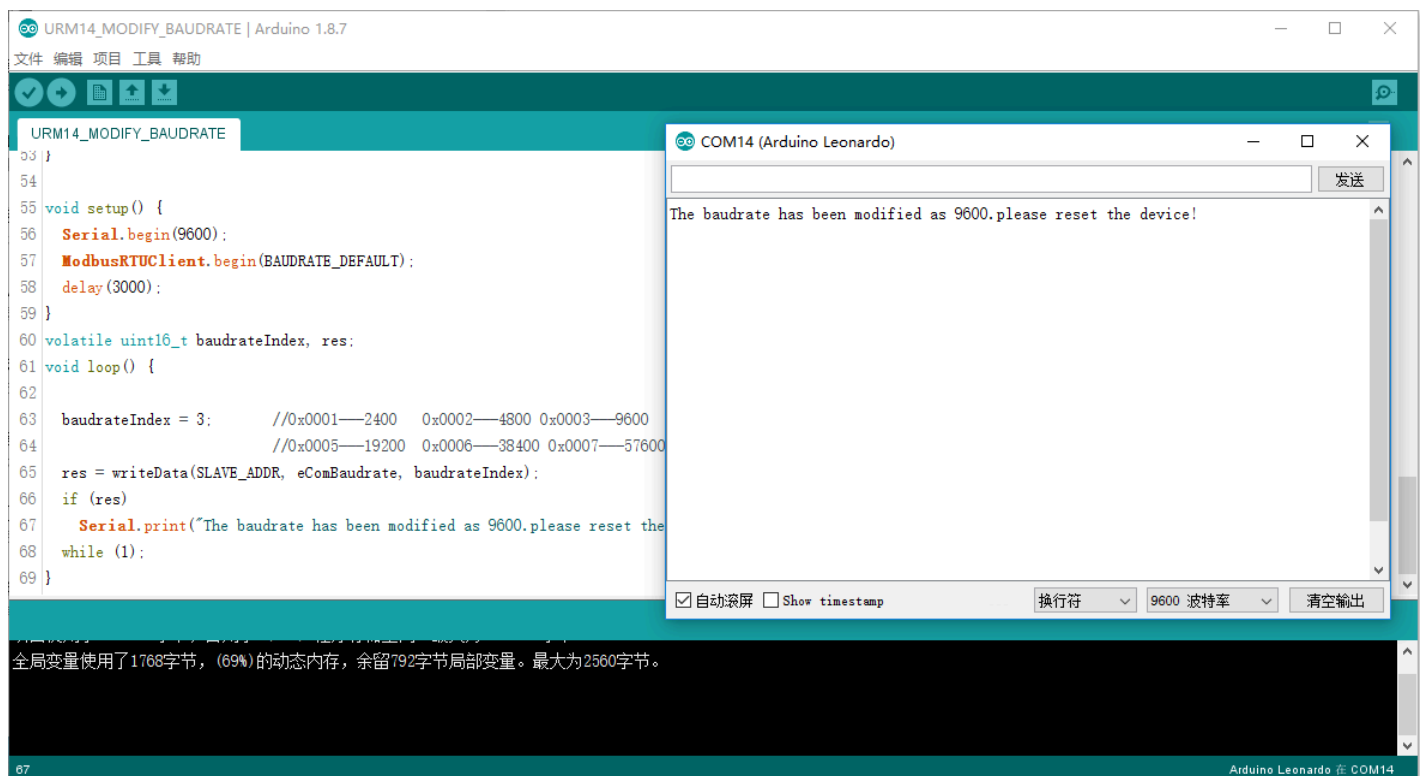
void setup() {
    Serial.begin(9600);
    ModbusRTUClient.begin(BAUDRATE_DEFAULT);
    delay(3000);
}
volatile uint16_t baudrateIndex, res;
void loop() {

```

```

baudrateIndex = 3;          //0x0001---2400  0x0002---4800 0x0003---9600  0x0004---19200
                             //0x0005---19200  0x0006---38400 0x0007---57600 0x0008---96000
res = writeData(SLAVE_ADDR, eComBaudrate, baudrateIndex); //Writes the new baudrate
if (res)
    Serial.print("The baudrate has been modified as 9600.please reset the device!");
while (1);
}

```



Detection Angle and Sensitivity

Normally, the detection area of an ultrasonic sensor is irregular and hard to define due to its physical characteristics, so the detection angle is indeterminate. In actual use, the detection area and sensitivity of this sensor prove to be smaller than those of other ultrasonic sensors. We used 2 different target obstacles to repeatedly test multiple sample products, and the reference detection area of the corresponding target is as follows:

- Detection area for PVC pipe of 7.5cm diameter
(<https://dfimg.dfrobot.com/nobody/wiki/719bac2c8caf9f5533ac3bcd6e4466a9.p>)

df)

- Detection area for 30cmx20cm plane plate
(<https://dfimg.dfrobot.com/nobody/wiki/41d5020e5b2ca8b2a5b231a8b07a91ad.pdf>)

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (<https://www.dfrobot.com/forum/>).

More Documents
