SKU:SEN0519 (https://www.dfrobot.com/product-2620.html)

(https://www.dfrobot.com/product-2620.html)

Introduction

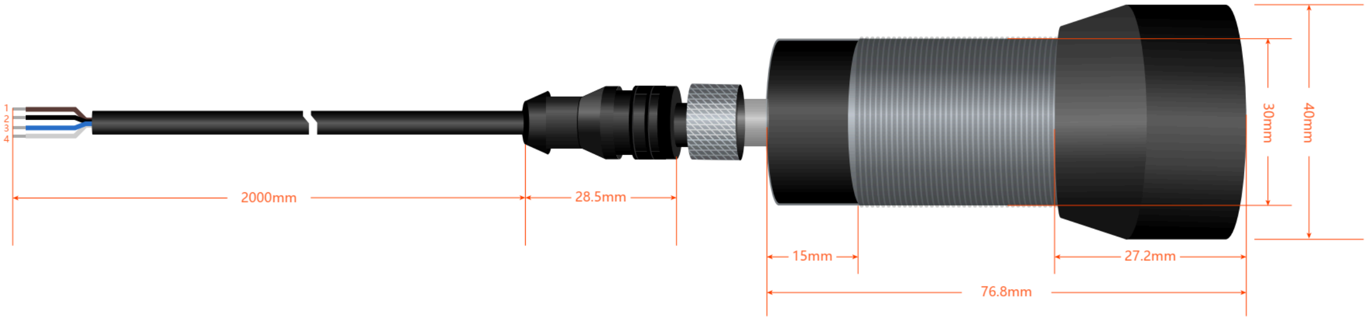
DFRobot URM15 is an ultrasonic ranging sensor with an IP65 waterproof probe that offers an effective measuring range of 30cm-500cm(test on flat wall). It uses the RS485 interface that follows the standard Modbus-RTU protocol for reliable data communication. The sensor supports revisable slave addresses and serial parameters, which can be conveniently used with all kinds of industrial controlling machines.
In addition, the URM15 comes with temperature compensation function. Users can select external or onboard temperature compensation to reduce the impact of ambient temperature on the measurement. The sensor adopts 75Khz ultrasonic transducer with diameter of 40mm, so it features a relatively small measuring angle and high sensitivity.

Specification

- Operating Voltage: 5-12V DC
- Max Instantaneous Current: 350mA
- Effective Measuring Range: 30cm - 500cm
- Distance Resolution: 10mm
- Distance Error: ±1%
- Temperature Resolution: 0.1℃
- Temperature Error: ±1℃
- Measuring Frequency: 10 Hz
- Operating Temperature: -10℃ to＋70℃
- Operating Humidity Range: RH<75%
- Sensor Acoustic Frequency: 75KHz±2%
- Directional Angle: 20±2°(-6dB)
- Communication: RS485(Modbus-RTU)

- Communication: RS485(Modbus-RTU)

Board Overview



Module Interface Line Sequence:

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

Register Description

Address	Number	Name	Read/Write	Data Range	Default	Data Description
0x00	1	Module PID register	R	0x0000-0xFFFF	0x0005	Product checl (detect module type)
0x01	1	Module VID register	R	0x0000-0xFFFF	0x0010	Version check (0x0010 represents V0.0.1.0)

Address	Number	Name	Read/Write	Data Range	Default	Data Description
0x02	1	Module address register	R/W	0x0001-0x00F7	0x000F	When the sensor address is unknown, write to the register through the broadcast address 0x00, at this time, the sensor wil not have data output **Save when powered off, take effect after restarting**

Address	Number	Name	Read/Write	Data Range	Default	Data Description
0x03	1	Serial parameter control register 1	R/W	0x0000-0xFFFF	0x0005	Module Baud Rate: 0x0001---2400 0x0003---9600 0x0004---14400 0x0005---19200 0x0006---38400 0x0007---57600 0x0008---115200 Other----115200 **Save when powered off, take effect after restarting**
0x04	1	Serial parameter control register 2	R/W	0x0000-0xFFFF	0x0001	Reserved (serial data format is fixe at: no parity bit, 1 stop bit, 8 data bits) **Save when powered off, take effect after**

						restarting

Address	Number	Name	Read/Write	Data Range	Default	Data Description
0x05	1	Distance register	R	0x0000-0xFFFF	0xFFFF	The distance value LSB measured by the module represents 0.1cm
0x06	1	Onboard temperature data Register	R	0x0000-0xFFFF	0x0000	The temperature value LSB measured by the onboard temperature sensor represents 0.1℃ (with unit symbol)
0x07	1	External temperature compensation data register	R/W	0x0000-0xFFFF	0x0000	Write ambien temperature data to this register for external temperature compensatior LSB represent 0.1℃ (with unit symbol)

Address	Number	Name	Read/Write	Data Range	Default	Data Description
0x08	1	Control register	R/W	0x0000-0xFFFF	0x0004	bit0: 0-use onboard temperature compensation function 1-use externa temperature compensation function (users need to write temperature data to external temperature compensation data register) bit1: 0-enable temperature compensation function 1-disable temperature compensation function bit2: 0-auto detection 1-passive detection bit3: In passive

Address	Number	Name	Read/Write	Data Range	Default	Data Description
						detection mode, write 1 to this bit, then it will measure distance once and the distance value can be read from distance register about 65ms later. In auto detection mode, this bit is reserved. This bit will be auto cleared when set to 1 **Save when powered off, take effect after**

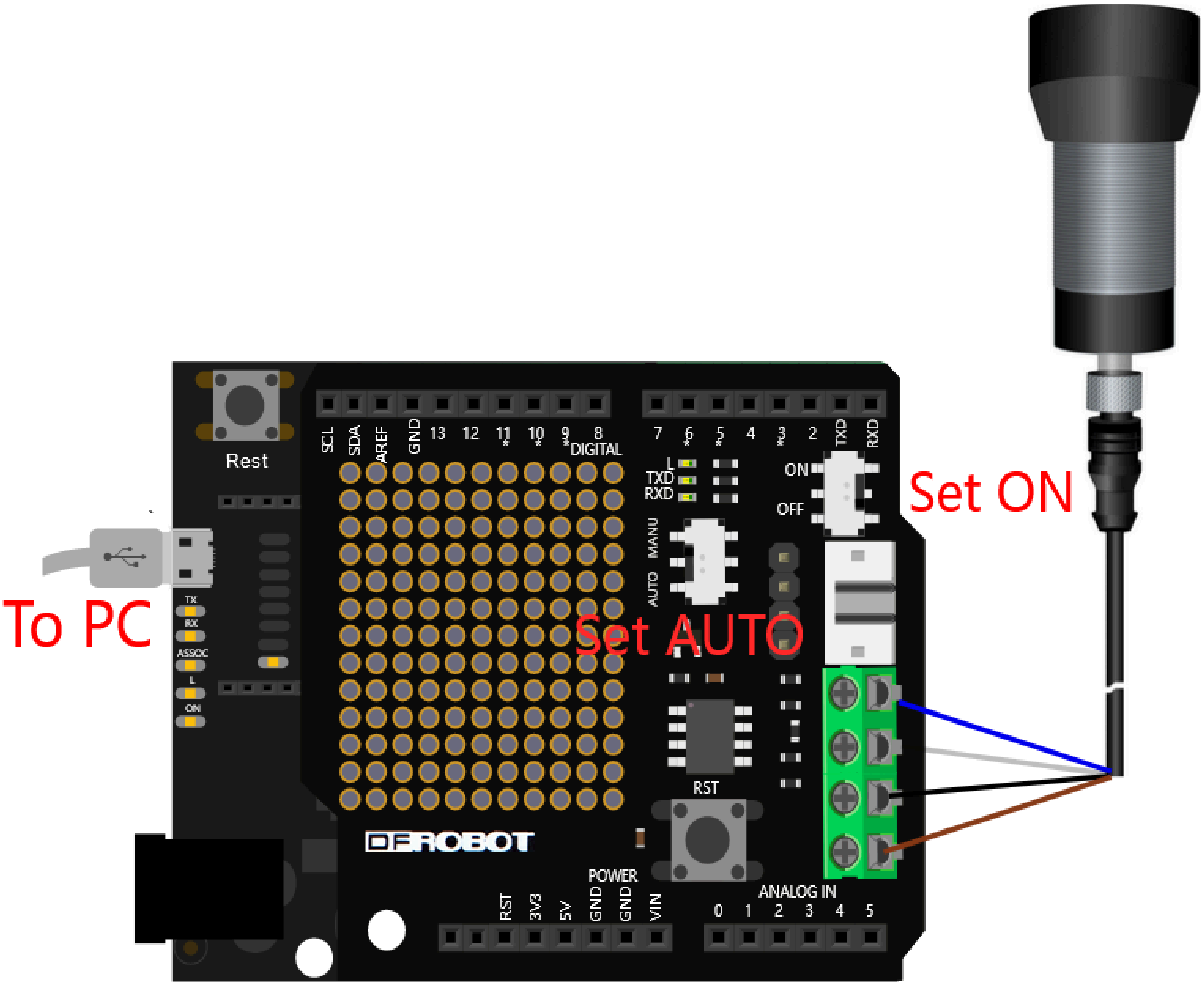
Register Read/Write Sample

Requirements

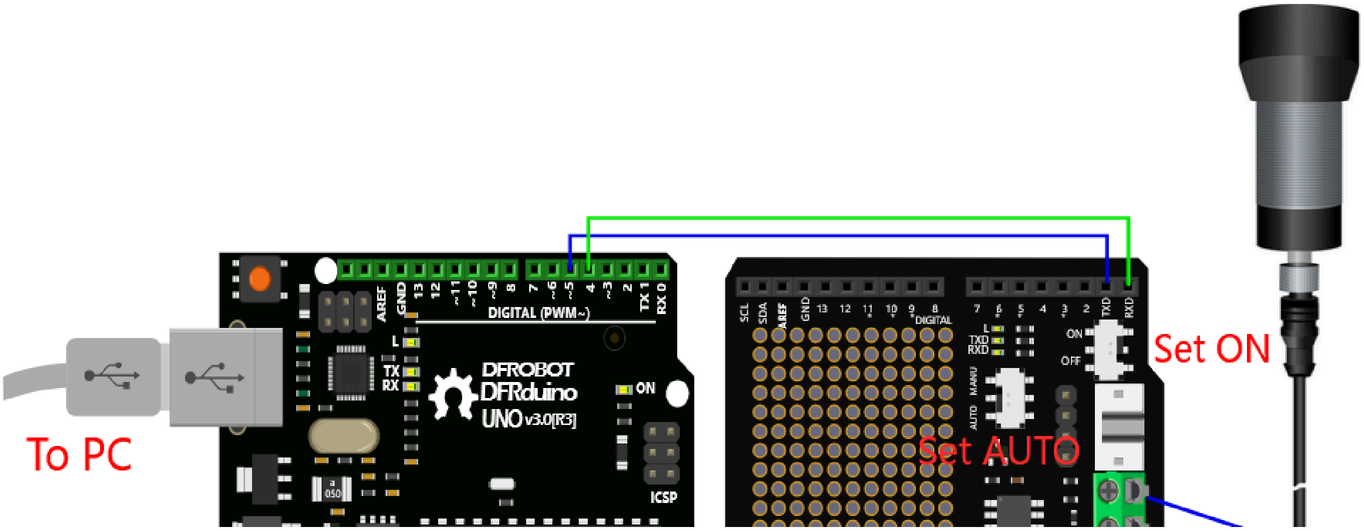
- **Hardware**
  - Arduino Leonardo (https://www.dfrobot.com/product-832.html) x 1
  - RS485 Shield for Arduino (https://www.dfrobot.com/product-1024.html) x 1
  - USB Data Cable x 1 (Connect the Arduino board to a computer via the USB cable)
- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Open **Library Manager(Ctrl+Shift+I)** in Arduino IDE, find and install **DFRobot_RTU**

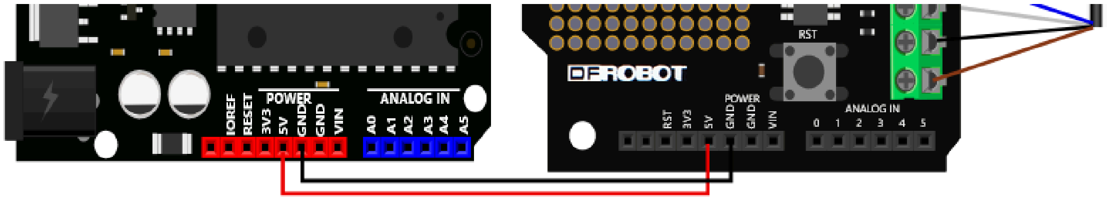
library.

- Diagram of Connecting to LEONARDO

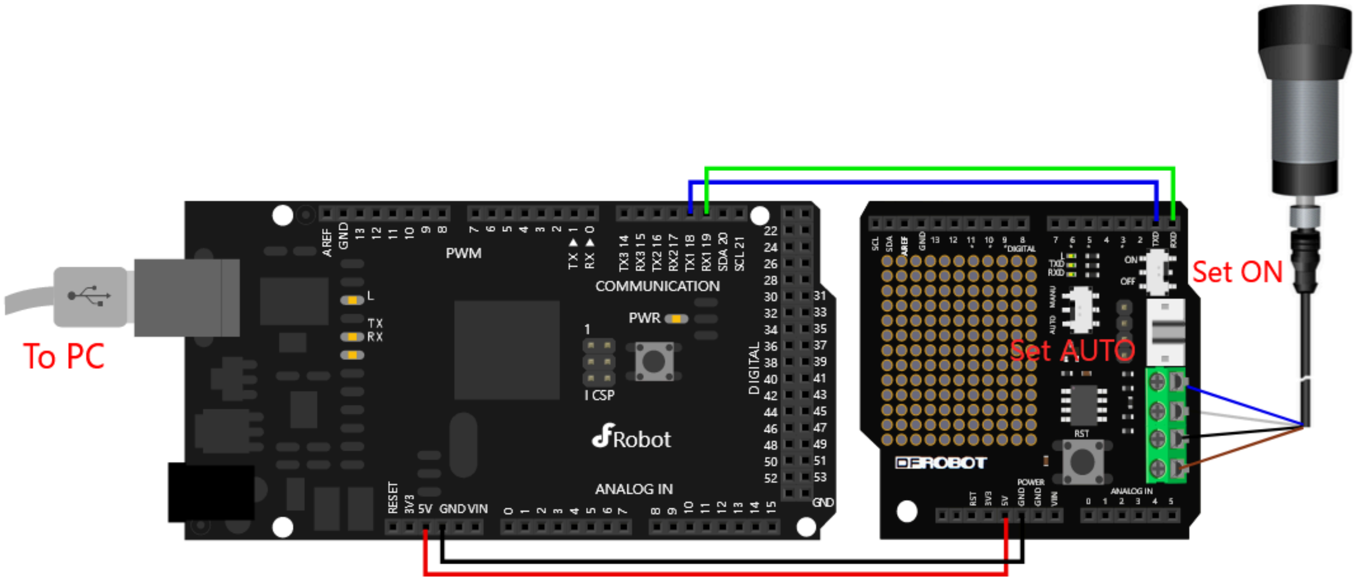


- Diagram of Connecting to UNO





- **Diagram of Connecting to MEGA**



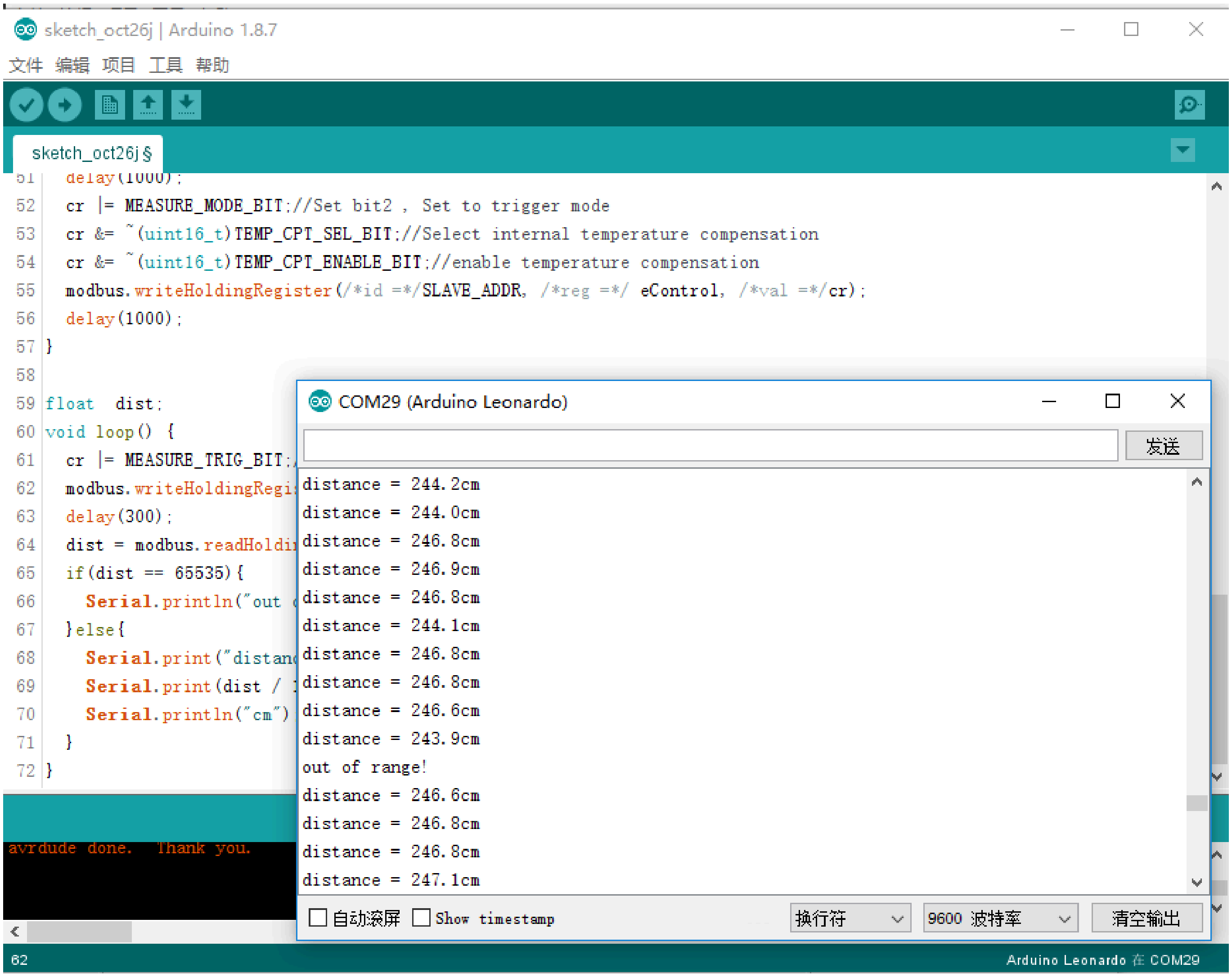
To PC

Set ON

Set AUTO

Read Detected Distance

```
/*****************************************************************************************
 * @This code tests the range finder function of the URM15 ultrasonic sensor
 * @brief Change device ID of modbus slave. Each modbus slave has a unique device ID numb
 * @ And there are two ways to change the device ID:
 * @n 1: If you don't know the device ID, you can change slave ID address by broadcast
 * @n address 0x00, this command will change the address of all the slaves on the bus to
 * @n changing address with 0x00, it is better to connect only one device on the bus)
 * @n 2: If you know the device ID, change it directly
 * @n note: To run this demo, you must know the serial port configuration of the device (
 * @n connected table
 * ---------------------------------------------------------------------------------------
 * sensor pin |             MCU                | Leonardo/Mega2560/M0 |    UNO    | ESP82
 *     VCC    |            3.3V/5V             |        VCC           |    VCC    |   VCC
 *     GND    |              GND               |        GND           |    GND    |   GND
 *     RX     |              TX                |     Serial1 RX1      |     5     |5/D6(T
 *     TX     |              RX                |     Serial1 TX1      |     4     |4/D7(R
 * ---------------------------------------------------------------------------------------
 * @copyright   Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence     The MIT License (MIT)
 * @ author : roker.wang@dfrobot.com
 * @ data   : 26.10.2021
 * @ version: 1.0
*****************************************************************************************
#include "DFRobot_RTU.h"
#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
#include <SoftwareSerial.h>
#endif

#define   SLAVE_ADDR                ((uint16_t)0x0F)

#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,
  eDistance,
  eInternalTempreture,
  eExternTempreture,
  eControl
}eRegIndex_t;//Sensor register index


#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
  SoftwareSerial mySerial(/*rx =*/4, /*tx =*/5);
  DFRobot_RTU modbus(/*s =*/&mySerial);
#else
  DFRobot_RTU modbus(/*s =*/&Serial1);
#endif

volatile uint16_t cr = 0;
void setup() {
  Serial.begin(9600);
  while(!Serial){                                                     //Waiting for USB S
  }

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
    mySerial.begin(19200);
#elif defined(ESP32)
  Serial1.begin(19200, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
#else
  Serial1.begin(19200);
#endif
  delay(1000);
  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
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eControl, /*val =*/cr);
  delay(1000);
}

float  dist;
void loop() {
  cr |= MEASURE_TRIG_BIT;//Set trig bit
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eControl, /*val =*/cr);
  delay(300);
  dist = modbus.readHoldingRegister(SLAVE_ADDR, eDistance);
  if(dist == 65535){
    Serial.println("out of range!");
  }else{
    Serial.print("distance = ");
    Serial.print(dist / 10, 1);
    Serial.println("cm");
  }
}
```



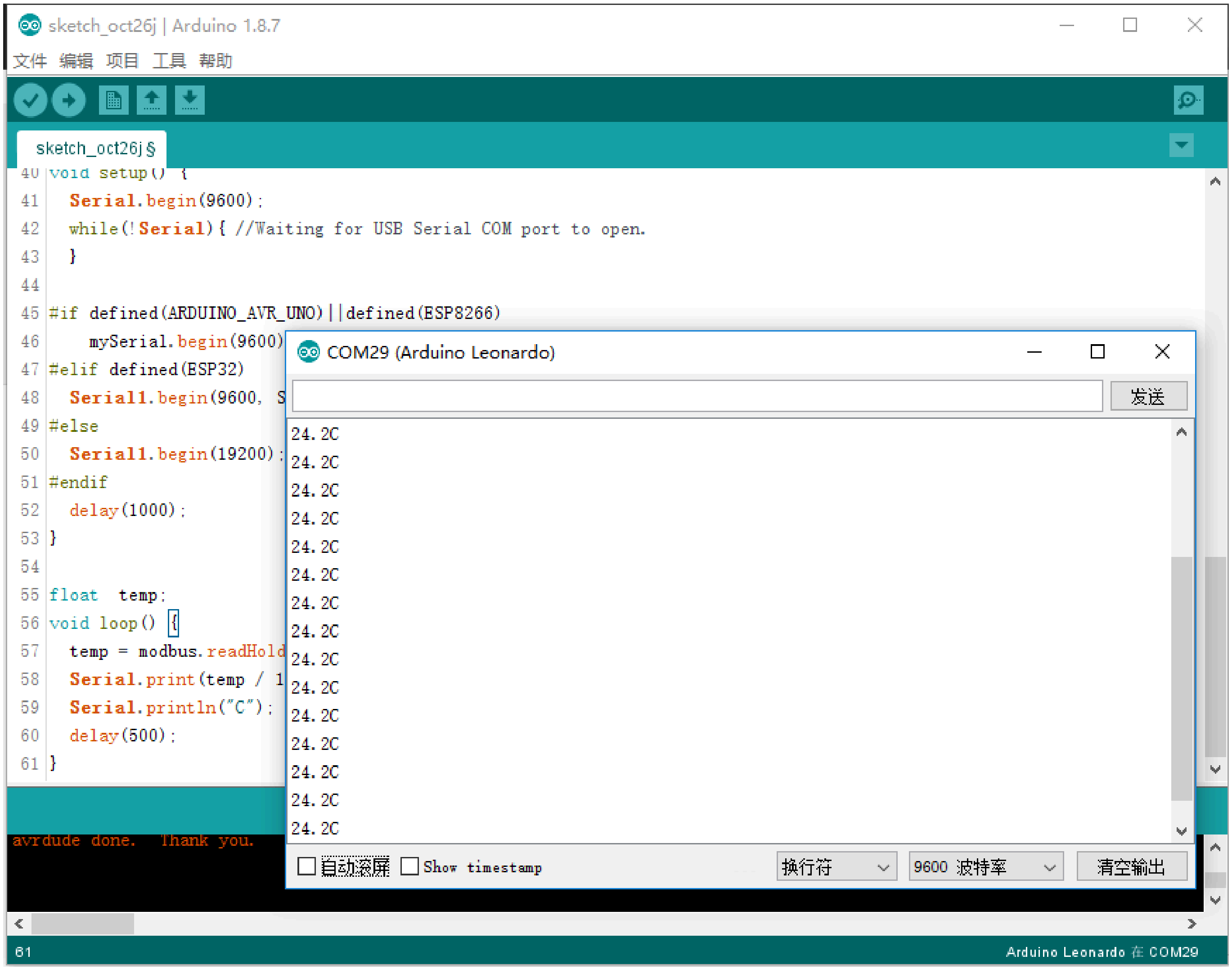
```
  delay(1000);
  cr |= MEASURE_MODE_BIT;//Set bit2 , Set to trigger mode
  cr &= ~(uint16_t)TEMP_CPT_SEL_BIT;//Select internal temperature compensation
  cr &= ~(uint16_t)TEMP_CPT_ENABLE_BIT;//enable temperature compensation
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eControl, /*val =*/cr);
  delay(1000);
}

float  dist;
void loop() {
  cr |= MEASURE_TRIG_BIT;/
  modbus.writeHoldingRegi
  delay(300);
  dist = modbus.readHoldi
  if(dist == 65535){
    Serial.println("out o
  }else{
    Serial.print("distan
    Serial.print(dist / 1
    Serial.println("cm")
  }
}
```

```
avrdude done.  Thank you.
```

```
distance = 244.2cm
distance = 244.0cm
distance = 246.8cm
distance = 246.9cm
distance = 246.8cm
distance = 244.1cm
distance = 246.8cm
distance = 246.8cm
distance = 246.6cm
distance = 243.9cm
out of range!
distance = 246.6cm
distance = 246.8cm
distance = 246.8cm
distance = 247.1cm
```

Read Onboard Temperature

```
/************************************************************************************
 * @This code tests the temperature measurement function of the URM15 ultrasonic sensor
 * @brief Change device ID of modbus slave. Each modbus slave has a unique device ID numb
 * @ And there are two ways to change the device ID:
 * @n 1: If you don't know the device ID, you can change slave ID address by broadcast
 * @n address 0x00, this command will change the address of all the slaves on the bus to
 * @n changing address with 0x00, it is better to connect only one device on the bus)
 * @n 2: If you know the device ID, change it directly
 * @n note: To run this demo, you must know the serial port configuration of the device (
 * @n connected table
 * ------------------------------------------------------------------------------------
 * sensor pin |             MCU                | Leonardo/Mega2560/M0 |    UNO    | ESP82
 *     VCC    |            3.3V/5V             |        VCC           |    VCC    |   VCC
 *     GND    |             GND                |        GND           |    GND    |   GND
 *     RX     |             TX                 |     Serial1 RX1      |     5     |5/D6(T
 *     TX     |             RX                 |     Serial1 TX1      |     4     |4/D7(R
 * ------------------------------------------------------------------------------------
 * @copyright   Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence     The MIT License (MIT)
 * @ author : roker.wang@dfrobot.com
 * @ data   : 26.10.2021
 * @ version: 1.0
*************************************************************************************
#include "DFRobot_RTU.h"
#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
#include <SoftwareSerial.h>
#endif

#define   SLAVE_ADDR                ((uint16_t)0x0F)

#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,
  eDistance,
  eInternalTempreture,
  eExternTempreture,
  eControl
}eRegIndex_t;//Sensor register index


#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
  SoftwareSerial mySerial(/*rx =*/4, /*tx =*/5);
  DFRobot_RTU modbus(/*s =*/&mySerial);
#else
  DFRobot_RTU modbus(/*s =*/&Serial1);
#endif

volatile uint16_t cr = 0;
void setup() {
  Serial.begin(9600);
  while(!Serial){ //Waiting for USB Serial COM port to open.
  }

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
    mySerial.begin(19200);
#elif defined(ESP32)
  Serial1.begin(19200, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
#else
  Serial1.begin(19200);
#endif
  delay(1000);
}

float  temp;
void loop() {
  temp = modbus.readHoldingRegister(SLAVE_ADDR, eInternalTempreture);
  Serial.print(temp / 10,1);
  Serial.println("C");
  delay(500);
}
```



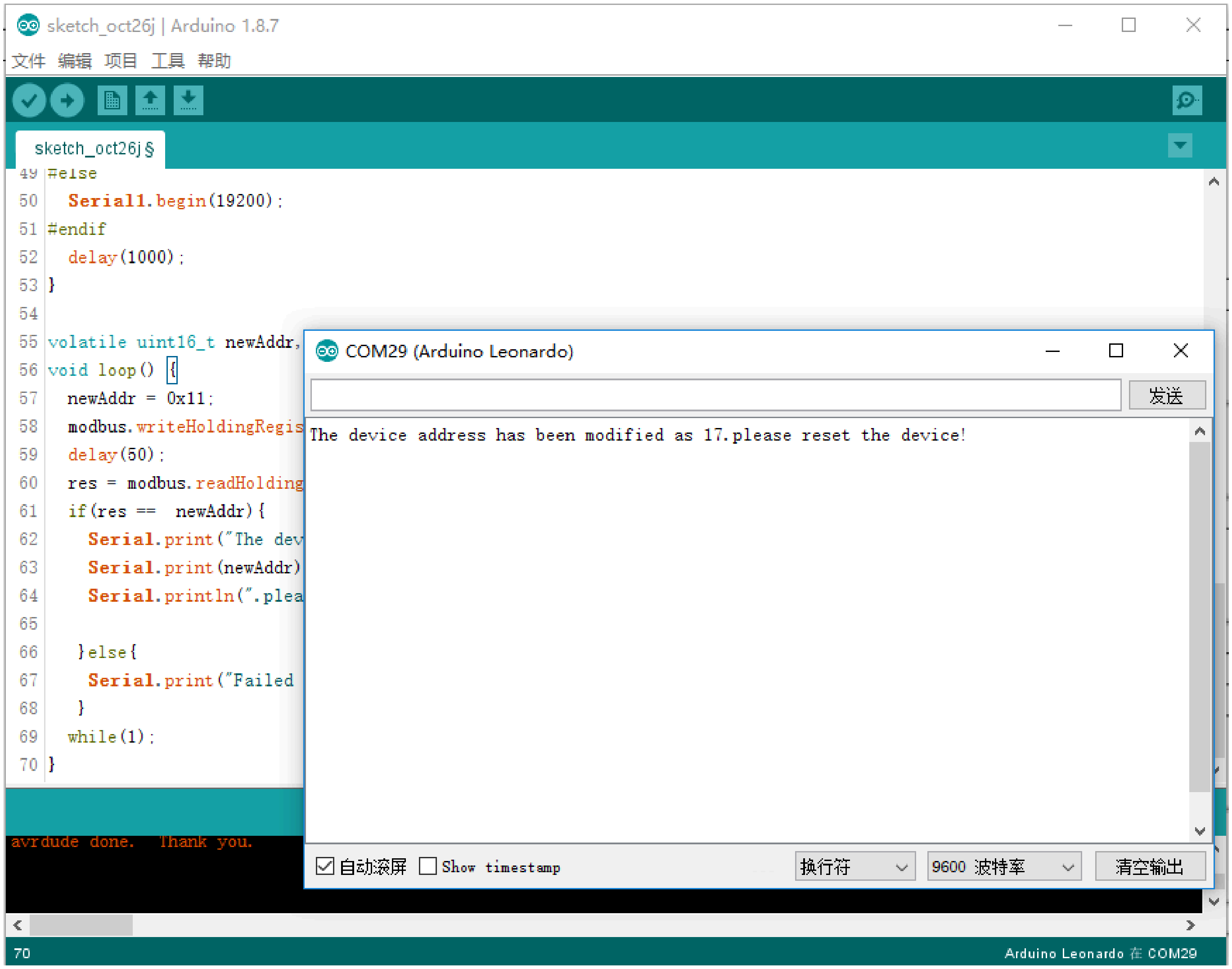
```
void setup() {
  Serial.begin(9600);
  while(!Serial){ //Waiting for USB Serial COM port to open.
  }

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
    mySerial.begin(9600)
#elif defined(ESP32)
  Serial1.begin(9600, S
#else
  Serial1.begin(19200);
#endif
  delay(1000);
}

float  temp;
void loop() {
  temp = modbus.readHold
  Serial.print(temp / 1
  Serial.println("C");
  delay(500);
}
```

```
avrdude done.  Thank you.
```

```
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
24.2C
```

Revise Module Address

```
/*******************************************************************************************
 * @This code tests the address modification function of the URM15 ultrasonic sensor
 * @brief Change device ID of modbus slave. Each modbus slave has a unique device ID numb
 * @ And there are two ways to change the device ID:
 * @n 1: If you don't know the device ID, you can change slave ID address by broadcast
 * @n address 0x00, this command will change the address of all the slaves on the bus to
 * @n changing address with 0x00, it is better to connect only one device on the bus)
 * @n 2: If you know the device ID, change it directly
 * @n note：To run this demo, you must know the serial port configuration of the device (
 * @n connected table
 * -----------------------------------------------------------------------------------------
 * sensor pin |             MCU                | Leonardo/Mega2560/M0 |    UNO    | ESP82
 *     VCC    |            3.3V/5V             |        VCC           |    VCC    |   VCC
 *     GND    |              GND               |        GND           |    GND    |   GND
 *     RX     |              TX                |     Serial1 RX1      |     5     |5/D6(T
 *     TX     |              RX                |     Serial1 TX1      |     4     |4/D7(R
 * -----------------------------------------------------------------------------------------
 * @copyright   Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence     The MIT License (MIT)
 * @ author : roker.wang@dfrobot.com
 * @ data   : 26.10.2021
 * @ version: 1.0
*******************************************************************************************
#include "DFRobot_RTU.h"
#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
#include <SoftwareSerial.h>
#endif

#define   SLAVE_ADDR                ((uint16_t)0x0F)

#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,
  eDistance,
  eInternalTempreture,
  eExternTempreture,
  eControl
}eRegIndex_t;//Sensor register index


#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
  SoftwareSerial mySerial(/*rx =*/4, /*tx =*/5);
  DFRobot_RTU modbus(/*s =*/&mySerial);
#else
  DFRobot_RTU modbus(/*s =*/&Serial1);
#endif

volatile uint16_t cr = 0;
void setup() {
  Serial.begin(9600);
  while(!Serial){                                                     //Waiting for USB S
  }

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
    mySerial.begin(19200);
#elif defined(ESP32)
  Serial1.begin(19200, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
#else
  Serial1.begin(19200);
#endif
  delay(1000);
}

volatile uint16_t newAddr, res;
void loop() {
  newAddr = 0x11;
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eAddr, /*val =*/newAddr);
  delay(50);
  res = modbus.readHoldingRegister(SLAVE_ADDR, eAddr);
  if(res ==  newAddr){
    Serial.print("The device address has been modified as ");
    Serial.print(newAddr);
    Serial.println(".please reset the device!");

   }else{
    Serial.print("Failed to change the sensor address!");
   }
  while(1);
}
```



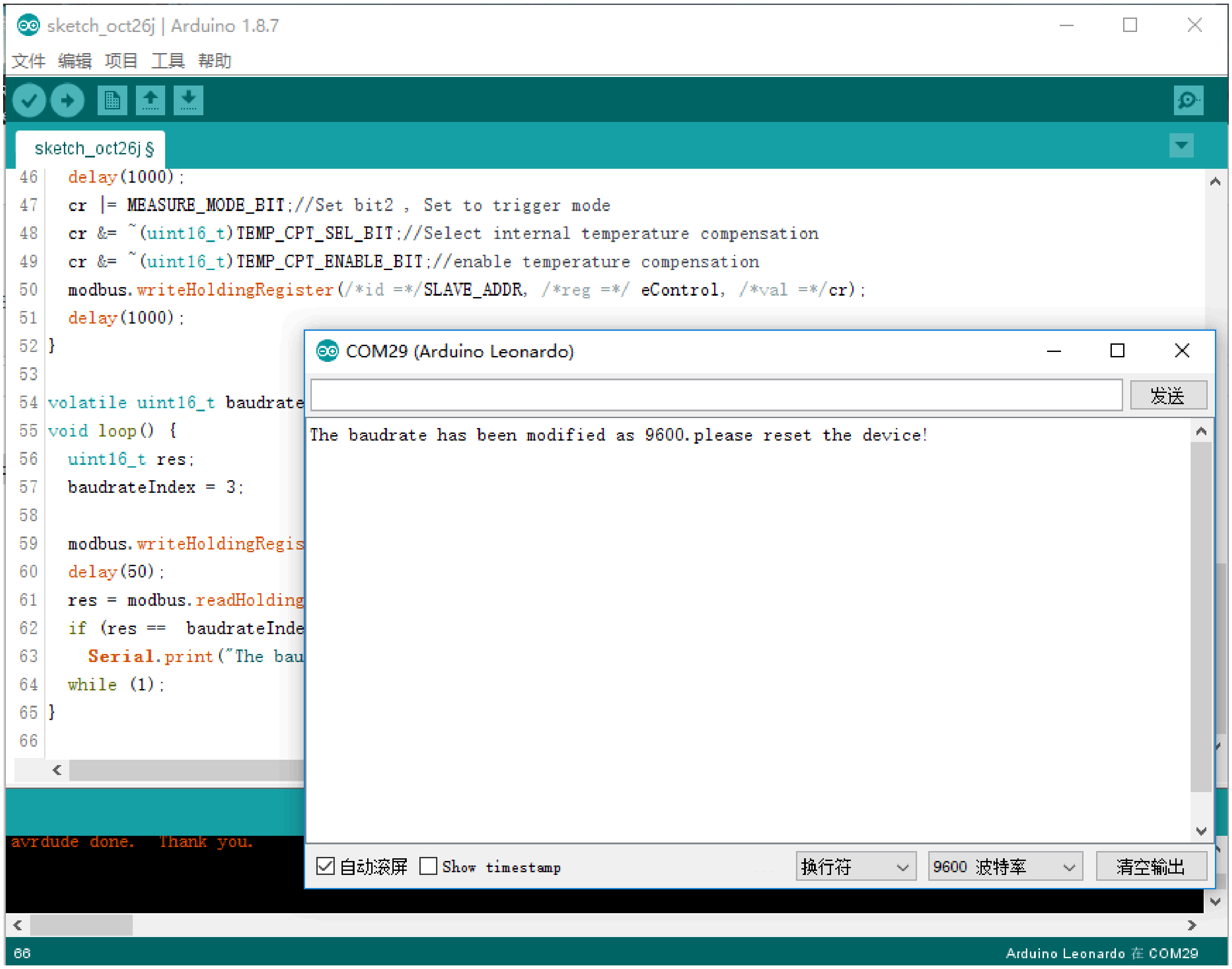
```
#else
  Serial1.begin(19200);
#endif
  delay(1000);
}

volatile uint16_t newAddr,
void loop() {
  newAddr = 0x11;
  modbus.writeHoldingRegis
  delay(50);
  res = modbus.readHolding
  if(res ==  newAddr){
    Serial.print("The dev
    Serial.print(newAddr)
    Serial.println(".plea

   }else{
    Serial.print("Failed
   }
  while(1);
}
```

```
avrdude done.  Thank you.
```

```
The device address has been modified as 17.please reset the device!
```

Revise Module Baud Rate

```
/*****************************************************************************************
 * @This code tests the baudrate modification function of the URM15 ultrasonic sensor
 * @brief Change device ID of modbus slave. Each modbus slave has a unique device ID numb
 * @ And there are two ways to change the device ID:
 * @n 1: If you don't know the device ID, you can change slave ID address by broadcast
 * @n address 0x00, this command will change the address of all the slaves on the bus to
 * @n changing address with 0x00, it is better to connect only one device on the bus)
 * @n 2: If you know the device ID, change it directly
 * @n note: To run this demo, you must know the serial port configuration of the device (
 * @n connected table
 * ---------------------------------------------------------------------------------------
 * sensor pin |             MCU                | Leonardo/Mega2560/M0 |    UNO    | ESP82
 *     VCC    |            3.3V/5V             |        VCC           |    VCC    |   VCC
 *     GND    |             GND                |        GND           |    GND    |   GND
 *     RX     |             TX                 |     Serial1 RX1      |     5     |5/D6(T
 *     TX     |             RX                 |     Serial1 TX1      |     4     |4/D7(R
 * ---------------------------------------------------------------------------------------
 * @copyright   Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @licence     The MIT License (MIT)
 * @ author : roker.wang@dfrobot.com
 * @ data   : 26.10.2021
 * @ version: 1.0
*****************************************************************************************

#include "DFRobot_RTU.h"
#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
#include <SoftwareSerial.h>
#endif

#define   SLAVE_ADDR                ((uint16_t)0x0F)

#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,
  eComParityStop,
  eDistance,
  eInternalTempreture,
  eExternTempreture,
  eControl

}eRegIndex_t;//Sensor register index

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
  SoftwareSerial mySerial(/*rx =*/4, /*tx =*/5);
  DFRobot_RTU modbus(/*s =*/&mySerial);
#else
  DFRobot_RTU modbus(/*s =*/&Serial1);
#endif

volatile uint16_t cr = 0;
void setup() {
  Serial.begin(9600);
  while(!Serial){                                                     //Waiting for USB S
  }

#if defined(ARDUINO_AVR_UNO)||defined(ESP8266)
    mySerial.begin(19200);
#elif defined(ESP32)
  Serial1.begin(19200, SERIAL_8N1, /*rx =*/D3, /*tx =*/D2);
#else
  Serial1.begin(19200);
#endif
  delay(1000);
  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
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eControl, /*val =*/cr);
  delay(1000);
}

volatile uint16_t baudrateIndex, res;
void loop() {
  uint16_t res;
  baudrateIndex = 3;       //0x0001---2400   0x0002---4800 0x0003---9600   0x0004---14400
                           //0x0005---19200  0x0006---38400 0x0007---57600 0x0008---11520
  modbus.writeHoldingRegister(SLAVE_ADDR, eComBaudrate, baudrateIndex);//Writes the new b
  delay(50);
  res = modbus.readHoldingRegister(SLAVE_ADDR, eComBaudrate);
  if (res ==  baudrateIndex)
    Serial.print("The baudrate has been modified as 9600.please reset the device!");
  while (1);
}
```



```
  delay(1000);
  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
  modbus.writeHoldingRegister(/*id =*/SLAVE_ADDR, /*reg =*/ eControl, /*val =*/cr);
  delay(1000);
}

volatile uint16_t baudrate
void loop() {
  uint16_t res;
  baudrateIndex = 3;

  modbus.writeHoldingRegis
  delay(50);
  res = modbus.readHolding
  if (res ==  baudrateInde
    Serial.print("The bau
  while (1);
}
```

```
The baudrate has been modified as 9600.please reset the device!
```

☑ 自动滚屏 □ Show timestamp 换行符 9600 波特率

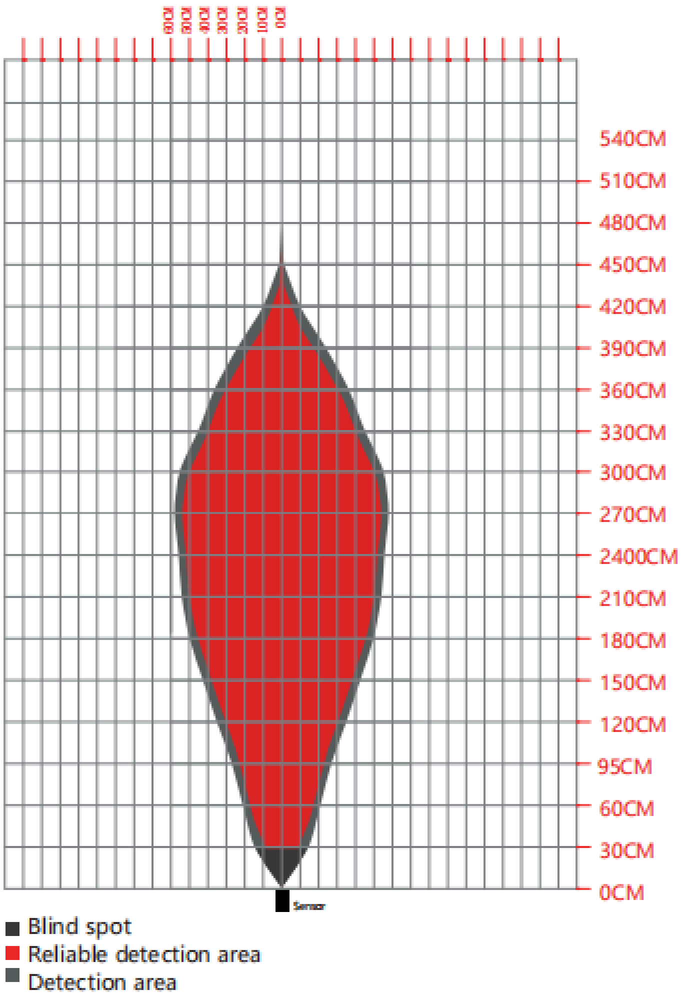
```
avrdude done.  Thank you.
```

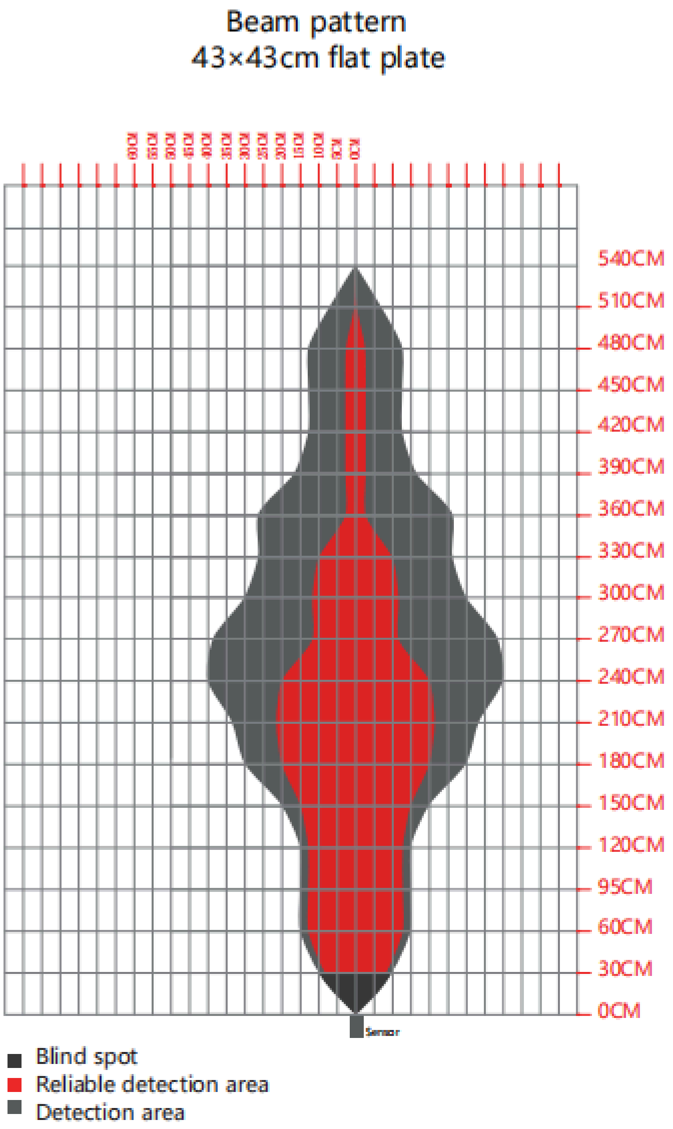
66 Arduino Leonardo 在 COM29

Detection Angle and Sensitivity

The detection area of an ultrasonic sensor is irregular and hard to define due to its physical characteristics. We used two kinds of reference target obstacles to repeatedly test many sample products. The reference detection area of the corresponding target is as follows:

Beam pattern
Diameter 7.5cm_PVC





FAQ

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

More Documents

Get **URM15 75KHZ Ultrasonic Sensor** (https://www.dfrobot.com/product-2620.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/distributor)

Turn to the Top

Turn to the Top