SKU:SEN0304 (https://www.dfrobot.com/product-1832.html)

Introduction

This is an ultrasonic distance sensor module with open dual probe. It adopts I2C communication and standard interface of Gravity PH2.0-4P vertical patch socket. The module is compatible with controllers with 3.3V or 5V logical level, such as Arduino and Raspberry Pi. The ultrasonic sensor comes with built-in temperature compensation, providing effective ranging within 2cm to 500cm. It offers resolution of 1cm and accuracy of ±1%. There are three measurement ranges designed for programs to select: 150cm, 300cm, 500cm. Please note that setting shorter range will cause shorter ranging cycle and lower sensitivity. You may need to set it according to the actual use.

Specification

- Supply Voltage: 3.3V ~ 5.5V DC

- Operating Current: 20mA
- Operating Temperature Range: -10℃ ~ ＋70℃
- Measurement Ranges: 2cm ~ 500cm (can be set)
- Resolution: 1cm
- Precision: 1%
- Direction Angle: 60°
- Frequency: 50Hz Max
- Dimension: 47mm × 22 mm/1.85″× 0.87″

Pinout

URM09 V2.0

NOTE: Compared with URM09 V1.0, the the latest URM09 V2.0 just improved the layout to improve its stability. And the dimension and function of V1.0 and V2.0 is the same.

improve its stability. And the dimension and function of V1.0 and V2.0 is the same.

Pin	Description
VCC	Power Supply(3.3V-5.5V)
GND	Ground
C	I2C SLC
D	I2C SDA

Tutorial

URM09 is a simple and practical ultrasonic sensor. It adopts I2C communication, which is very convenient to communicate with other boards that is equipped with I2C interface.

URM09 Ultrasonic Sensor(Gravity I^{2}C)(V1.0) Register

Register(8bit)	Name	R/W	Data Range	Default Value	Description

Register(8bit)	Name	R/W	Data Range	Default Value	Description
0×00	Device Address	R/W	0x08-0x77	0x11	I2C salve address, the default address is 0x11. If the address is changed, the new address will be valid after repowering the module.
0×01	Product ID	R		0x01	Used for product check
0×02	Version Number	R		0x10	Used for Version check(0x10 means V1.0)
0×03	Distance Value High-	R	0x00-0xFF	0x00	LSB represents 1CM. e.g. 0x64 represents 100CM

order bits

Register(8bit)	Name	R/W	Data Range	Default Value	Description
0×04	Distance Value Low-order bits	R	0x00-0xFF	0x00	LSB represents 1CM. e.g. 0x64 represents 100CM
0×05	Temperature Value High-order bits	R	0x00-0xFF	0x00	10 times amplified value based on the real temperature. e.g. if the readout value is 0x00fe, the real temperature value should be 0x00fe / 10 = 25.4℃

Register(8bit)	Name	R/W	Data Range	Default Value	Description
0×06	Temperature Value Low-order bits	R	0x00-0xFF	0x00	10 times amplified value based on the real temperature. e.g. if the readout value is 0x00fe, the real temperature value should be 0x00fe / 10 = 25.4℃
0×07	Configure Registers	R/W		0x00	Bit7(control bit in ranging mode) 0: passive measurement, send ranging command once, the module ranges the distance once

Register(8bit)	Name	R/W	Data Range	Default Value	Description
					and store the measured value into the distance register. 1: automatic measurement mode, the module keeps ranging distance and updating the distance register all the time. Bit6: save Bit5-bit4(the maximum ranging distance bit that can be set) 00:150CM(the ranging cycle is about 20MS) 01:300CM(the ranging cycle is about 30MS) 10:500CM(the ranging cycle is about 40MS)

Register(8bit)	Name	R/W	Data Range	Default Value	Description
0×08	Command Registers	R/W		0x00	Writing 0X01to this register under passive measurement mode and the module ranges distance once. The write data is invalid under automatic measurement mode.

Connection Diagram

Connect the module to UNO via I2C interface, shown as below.

URM09 Wiring

- URM09 Wiring

Distance and Temperature Measurement

The default measurement mode is passive mode. Send ranging command by write register, and then the module will start ranging once after receiving the command(ranging cycle is related to measurement range). When finished the measurement, the read distance register could obtain a distance value. When measuring the temperature, just read the temperature register and then the measured value can be available after simple processing.

Sample Code

```
// # Editor     : roker
// # Date       : 15.11.2018

// # Product name: URM V5.0 ultrasonic sensor
// # Product SKU : SEN0304
// # Version     : 1.0

#include <Wire.h>

unsigned char txbuf[10] = {0};
unsigned char rxbuf[10] = {0};


typedef enum {

  SLAVEADDR_INDEX = 0,
  PID_INDEX,
  VERSION_INDEX ,
  DIST_H_INDEX,
  DIST_L_INDEX,

  TEMP_H_INDEX,

  TEMP_L_INDEX,

  CFG_INDEX,
  CMD_INDEX,
  REG_NUM

} regindexTypedef;

#define    MEASURE_MODE_PASSIVE    (0x00)
#define    MEASURE_RANG_500        (0x20)
#define    CMD_DISTANCE_MEASURE    (0x01)



unsigned char addr0 = 0x11;

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
  Serial.begin(9600); // join i2c bus (address optional for master)
  txbuf[0] =  (MEASURE_MODE_PASSIVE | MEASURE_RANG_500);//the measurement mode is set to p
  i2cWriteBytes(addr0, CFG_INDEX , &txbuf[0], 1 );//
  delay(100);
}

void i2cWriteBytes(unsigned char addr_t, unsigned char Reg , unsigned char *pdata, unsigne
{
  Wire.beginTransmission(addr_t); // transmit to device #8
  Wire.write(Reg);              // sends one byte

  for (uint8_t i = 0; i < datalen; i++) {
    Wire.write(*pdata);
    pdata++;
  }

  Wire.endTransmission();    // stop transmitting
}

void i2cReadBytes(unsigned char addr_t, unsigned char Reg , unsigned char Num )
{
  unsigned char i = 0;
  Wire.beginTransmission(addr_t); // transmit to device #8
  Wire.write(Reg);              // sends one byte
  Wire.endTransmission();    // stop transmitting
  Wire.requestFrom(addr_t, Num);
  while (Wire.available())   // slave may send less than requested
  {
    rxbuf[i] = Wire.read();
    i++;
  }


}



unsigned char i = 0, x = 0;
void loop() {
  int16_t  dist, temp;
  txbuf[0] = CMD_DISTANCE_MEASURE;

  i2cWriteBytes(addr0, CMD_INDEX , &txbuf[0], 1 );//write register, send ranging command
  delay(100);

  i2cReadBytes(addr0, DIST_H_INDEX , 2 );//read distance register
  dist = ((uint16_t)rxbuf[0] << 8) + rxbuf[1];

  i2cReadBytes(addr0, TEMP_H_INDEX , 2 );//read temperature register
  temp = ((uint16_t)rxbuf[0] << 8) + rxbuf[1];

  Serial.print(dist, DEC);
```

```
    Serial.print("cm");
    Serial.print("------");

    Serial.print((float)temp / 10, 1);
    Serial.println("℃");


 }
```

Result

Result

I2C Address Setup

The default I2C address of the module is 0x11, and users could change it according to the actual use(the new address would only be valid after repowering the module).

Sample Code

```
// # Editor     : roker
// # Date       : 15.11.2018

// # Product name: URM V5.0 ultrasonic sensor
// # Product SKU : SEN0304
// # Version     : 1.0

#include <Wire.h>

unsigned char txbuf[10] = {0};
unsigned char rxbuf[10] = {0};
unsigned char addr0 = 0x11;

typedef enum {

  SLAVEADDR_INDEX = 0,
  PID_INDEX,
  VERSION_INDEX ,
  DIST_H_INDEX,
  DIST_L_INDEX,

  TEMP_H_INDEX,

  TEMP_L_INDEX,

  CFG_INDEX,
  CMD_INDEX,
  REG_NUM

} regindexTypedef;

#define    MEASURE_MODE_PASSIVE    (0x00)
#define    MEASURE_RANG_500        (0x20)
#define    CMD_DISTANCE_MEASURE    (0x01)

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
  Serial.begin(9600);
  txbuf[0] =  0x12;
  delay(100);
}

```

```
void i2cWriteBytes(unsigned char addr_t, unsigned char Reg , unsigned char *pdata, unsigne
{
  Wire.beginTransmission(addr_t); // transmit to device #8
  Wire.write(Reg);              // sends one byte


  for (uint8_t i = 0; i < datalen; i++) {
    Wire.write(*pdata);
    pdata++;
  }

  Wire.endTransmission();    // stop transmitting
}

void i2cReadBytes(unsigned char addr_t, unsigned char Reg , unsigned char Num )
{
  unsigned char i = 0;
  Wire.beginTransmission(addr_t); // transmit to device #8
  Wire.write(Reg);              // sends one byte
  Wire.endTransmission();    // stop transmitting
  Wire.requestFrom(addr_t, Num);
  while (Wire.available())   // slave may send less than requested
  {
    rxbuf[i] = Wire.read();
    i++;
  }

}

void loop() {

  uint8_t addr;

  i2cReadBytes(addr0, SLAVEADDR_INDEX , 1 );//read slave address register
  addr = rxbuf[0];
  Serial.print("the old i2c slave address is ");
  Serial.print("0x");
  Serial.println(addr, HEX);
  delay(10);

  i2cWriteBytes(addr0, SLAVEADDR_INDEX , &txbuf[0], 1 ); //write the new address 0×12 to a
  delay(100);
  i2cReadBytes(addr0, SLAVEADDR_INDEX , 1 );//read slave address register
  addr = rxbuf[0];
  Serial.print("the new i2c slave address is ");
  Serial.print("0x");
  Serial.println(addr, HEX);
  while(1);
}
```

Result

Address Setup

I2C Address Search

You may forget the I2C adress after setup, but you can search the I2C adress of the module through the following code. Shown as below.

```
#include <Wire.h>

void setup()
{
  Serial.begin(9600); delay(150);
  Serial.println("nI2C Scanner");
  Wire.begin();
}

void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for (address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);

    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error == 4)
    {
      Serial.print("Unknow error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }

  }
  if (nDevices == 0)
    Serial.println("No I2C devices found");
  else
    Serial.println("done");


  delay(1000);           // wait 5 seconds for next scan
}
```

Result

I2C Address

FAQ

Q&A	Some general Arduino Problems/FAQ/Tips
A	Please click the topic link (https://www.dfrobot.com/forum/viewtopic.php?f=8&t=1869&p=8624#p8624) on DFRobot Forum.
A	For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**

	(https://www.dfrobot.com/forum/).
Q&A	**Some general Arduino Problems/FAQ/Tips**

More Documents

- V1.0 Dimension Diagram
- V2.0 Dimension Diagram
- Find more Ultrasonic Sensors on DFRobot (https://www.dfrobot.com/category-55.html)

DFshopping_car1.png Get **URM09 Ultrasonic Sensor(Gravity I^2C)** (https://www.dfrobot.com/product-1832.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/index.php?route=information/distributorslogo)

Turn to the Top