URM37 V3.2 Ultrasonic Sensor (SKU:SEN0001)

From Robot Wiki

Contents

- 1 Introduction
- 2 Specification
  - 2.1 Compare with other ultrasonic sensor
- 3 Hardware requierments
- 4 Tools used
- 5 Software
- 6 Working Mode Selection
  - 6.1 Mode 1: Serial passive control mode
  - 6.2 Jumper setting for RS232 and TTL output
  - 6.3 Module test
  - 6.4 Mode 2: Autonomous trigger mode
  - 6.5 Mode 3: PWM passive control mode
  - 6.6 The sketch for PWM passive control mode
- 7 Serial control protocol
- 8 Arduino Sketch
- 9 Resources

Introduction

URM37 V3.2 Ultrasonic Sensor uses an industrial level AVR processor as the main processing unit. It comes with a temperature correction which is very unique in its class.

URM37 V3.2 Ultrasonic Sensor Manual - Rev 2.2

Specification

- Power: +5V
- Current: <20mA
- Working temperature: -10℃～+70℃
- Detecting range: 4cm-5m
- Resolution: 1cm
- Interface: RS232 (TTL), PWM
- Servo control: One servo control output
- Operating Mode: Serial; (PWM) passive control mode; Autonomous Mode; On/OFF Mode
- Temperature sensor: 12 bits reading from serial port
- Size: 22mm × 51 mm
- Weight: 30g

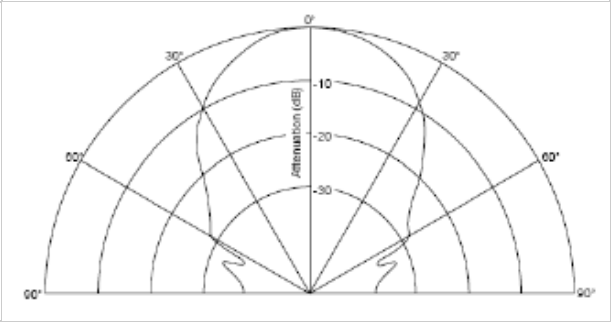


Figure 1: URM37 V3.2 Beam Width

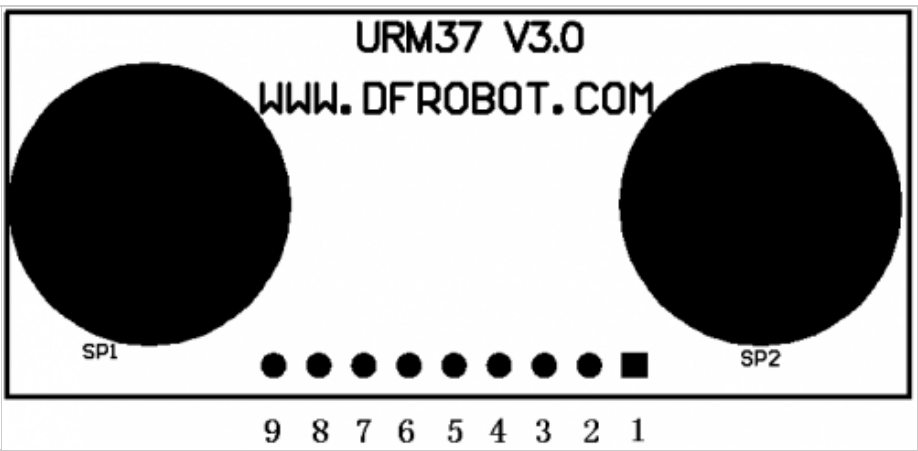


Figure 2: URM37 V3.2 Pin Definition

1. **+VCC** - +5V Power
2. **GND** - Ground
3. **RST** - Reset
4. **PWM** - PWM Output 0—25000US，Every 50US represent 1cm
5. **MOTO** - Servo control signal output
6. **COMP/TRIG**

    COMP - On/OFF mode, when the detecting distance is smaller than a pre-set value, this pin pulls low.
    TRIG - PWM or RS232 trigger pin

7. **NC**
8. **RXD** - RS232,TTL communication
9. **TXD** - RS232,TTL communication

Hardware requierments

1. 1×URM37 V3.2 Ultrasonic Sensor
2. 1×Arduino Microcontroller
3. 1×IO Expansion Shield For Arduino(V5)
4. 1×USB cable

Tools used

- 4×jumper wire

Software

- Arduino IDE

Working Mode Selection

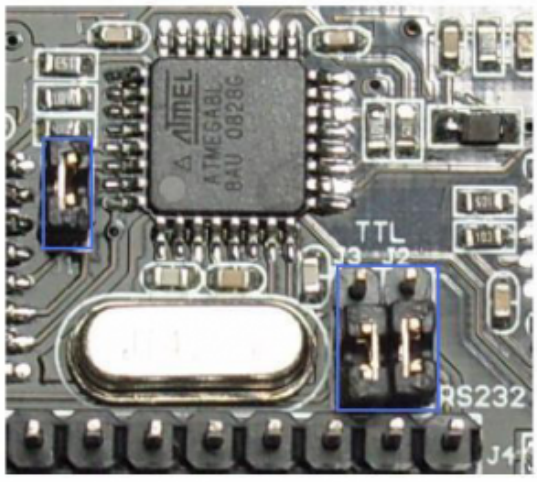
The working mode can be changed by writing 0x00, 0x01 or 0x02 to EEPROM through serial port.

Mode 1: Serial passive control mode

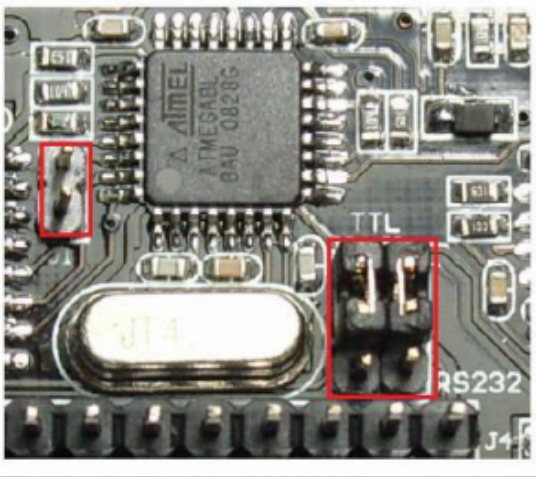
Under this mode, the sensor is always waiting for command from serial port. Every time it receives a command, it will return the distance and wait for the next command. The degree in the command will be used to control a servo motor to rotate corresponding degree. Please note that this mode is always on. It can not be switch on or off.

Jumper setting for RS232 and TTL output

The selection of RS232 or TTL output level is switched by changing three jumpers (J1, J2, J3). A diagram below illustrates the setting:



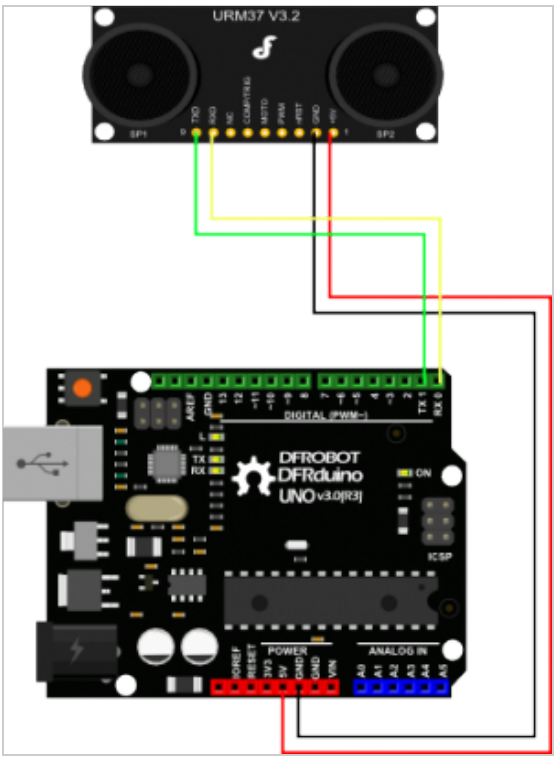
RS232 Mode



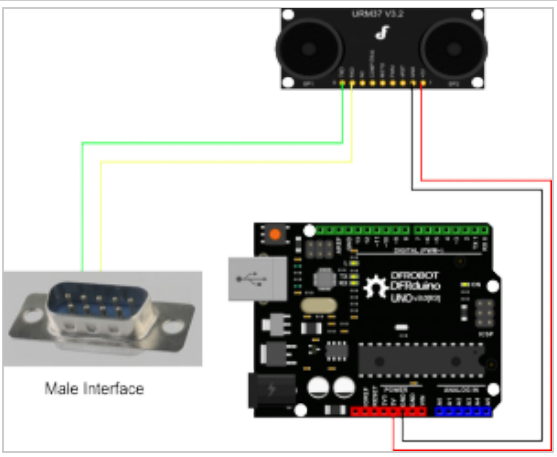
TTL Mode

Warning: Do not connect to TTL MCU when the output mode is set to RS232, doing so will permanently damage the unit.

This feature is only available for Rev2 and after. If there are no jumpers on the back of the sensor, the sensor is Rev1 and hence not supporting this feature.



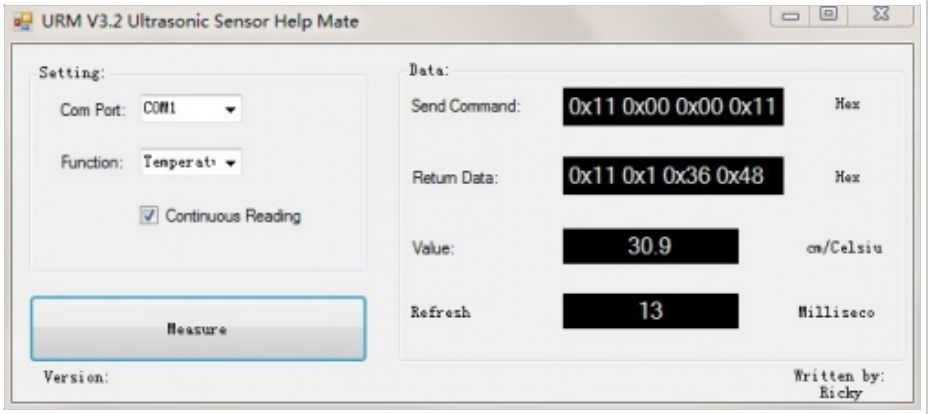
TTL Mode Connection Diagram



RS232 Mode Connection Diagram

Module test

After you have connected the module according to the diagram,you can use our "URMV3.2HelpMate (http://www.dfrobot.com/image/data/SEN0001/URMV3.2HelpMate.rar)" to test the module online



URMV3.2HelpMate

The usage of the software is very simple: ensure that there is no other software on the computer takes up the serial port, and then running mate, select the COM Port, and choose the parameter what you want to measure,and choose the "Continuous Reading ".Click "Measure" it will measure the temperature and the distance.

Mode 2: Autonomous trigger mode

Under this mode, the sensor will make a sensor reading every 25ms and compare the reading with a threshold (pre-set, user is able to define this value by writing EEPROM), if the reading is equal or smaller than the threshold, pin COMP/TRIG will have low output. In the meantime, pin PWM will output the distance reading, every 50us low level stands for 1cm, by counting the number of these pulses, the distance can be calculated. This mode can be simply used as an ON/OFF switch.

First you need to write the desired distance threshold into the sensor module. Using the serial port and the following code.

The way to write its EEPROM can be found in the code below. And the distance is stored in address 0x00 and 0x01 in cm, that's to say if the threshold you want is 15cm, you should write a 0x0f（as 0x0f is equal to 15 ）into address 0x00 and 0x00 in address 0x01， and never forget that once you change the distance threshold,the sum is also needed to be corrected. Details for the data is in the table below. Briefly, works like this cmd1<cmd, address, data, checksum>

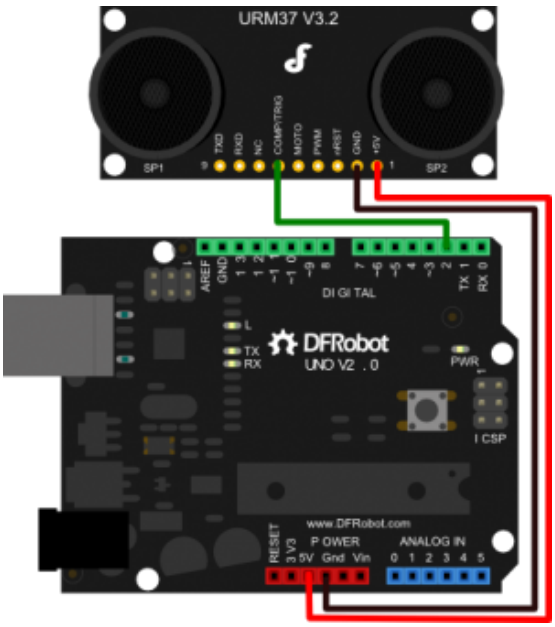
checkSum = Low 8 bit of the sum of command+data0+data1

```
?
1 // # Editor    :Holiday from DFRobot
2 // # Data      :30.05.2013
3
4 // # Product name:ultrasonic scanner
5 // # Product SKU:SEN0001
6 // # Version :  3.2
7
8 // # Description:
9 // # This sample shows how to use the Autonomous trigger mode by writing its EEPROM
10
11// # Connection:
12// #       Pin 1 VCC (URM V3.2) -> VCC (Arduino)
13// #       Pin 2 GND (URM V3.2) -> GND (Arduino)
14// #       Pin 6 COMP/TRIG (URM V3.2) -> Pin 2 (Arduino)
15// #
16
17int cmmd1[]={
18  0x44,0x00,0x10,0x54};//low byte stored in the sensor for the distance threshold.
19int cmmd2[]={
20  0x44,0x01,0x00,0x45};//high byte, write 0x0010 into address 0x01 and 0x00,so the threshold is set to 16cm
21int cmmd3[]={
22  0x44,0x02,0xaa,0xf0};// Autonomous mode. write 0xaa into address 0x02
```

```
//int cmmd3[]={
//  0x44,0x02,0xbb,0x01};  // PWM mode. write 0xbb into address 0x02
int i;

void setup(){
  Serial.begin(9600);                         // Sets the baud rate to 9600
  A_Mode_Setup();                             //PWM mode setup function
}

void loop()
{
}

void A_Mode_Setup(){
  //write the data into the URM37 EEPROM
  for (i=0;i<4;i++)
    Serial.write(cmmd3[i]);
  delay(200);

  for (i=0;i<4;i++)
    Serial.write(cmmd1[i]);
  delay(200);

  for (i=0;i<4;i++)
    Serial.write(cmmd2[i]);
  delay(200);

}
```

Remember to unplug the serial pins from Arduino before uploading your code.

After the code is uploaded, press reset on your Arduino board to confirm it is written on the module. Then you can connect your sensor with the following wiring system. And just read the pin for changes, when the threshold distance is reached.



?

```
int pin = 13;
volatile int state = LOW;

void setup(){
```

```
  pinMode(pin, OUTPUT);
  attachInterrupt(0, user_diy, CHANGE);          //The ON/OFF switch can be used as a singal of interruption
}

void loop(){
  digitalWrite(pin, state);
}

void user_diy()              //user can give your own code in this interruput function
{
  state = !state;
}
```

Mode 3: PWM passive control mode

Under this mode, a low pull on pin COMP/TRIG will trigger a sensor reading. The width of the pulse is proportional to the servo rotating degree. After a successful sensor reading, Pin PWM will output pulses, every 50us represents 1cm. If the reading is invalid, a 50000us pulse will be returned.

The sketch for PWM passive control mode

```
?
// # Editor    :Jiang from DFRobot
// # Data      :18.09.2012

// # Product name:ultrasonic scanner
// # Product SKU:SEN0001
// # Version :  0.2

// # Description:
// # The Sketch for scanning 180 degree area 4-500cm detecting range

// # Connection:
// #       Pin 1 VCC (URM V3.2) -> VCC (Arduino)
// #       Pin 2 GND (URM V3.2) -> GND (Arduino)
// #       Pin 4 PWM (URM V3.2) -> Pin 3 (Arduino)
// #       Pin 6 COMP/TRIG (URM V3.2) -> Pin 5 (Arduino)
// #
int URPWM = 3; // PWM Output 0—25000US，Every 50US represent 1cm
int URTRIG=5; // PWM trigger pin

unsigned int Distance=0;
uint8_t EnPwmCmd[4]={0x44,0x02,0xbb,0x01};    // distance measure command

void setup(){                                 // Serial initialization
  Serial.begin(9600);                         // Sets the baud rate to 9600
  PWM_Mode_Setup();
}

void loop()
{
 PWM_Mode();
 delay(20);
}                       //PWM mode setup function

void PWM_Mode_Setup(){
```

```
35  pinMode(URTRIG,OUTPUT);                    // A low pull on pin COMP/TRIG
36  digitalWrite(URTRIG,HIGH);                 // Set to HIGH
37
38  pinMode(URPWM, INPUT);                     // Sending Enable PWM mode command
39
40  for(int i=0;i<4;i++){
41      Serial.write(EnPwmCmd[i]);
42   }
43}
44
45void PWM_Mode(){                             // a low pull on pin COMP/TRIG  triggering a sensor reading
46    digitalWrite(URTRIG, LOW);
47    digitalWrite(URTRIG, HIGH);              // reading Pin PWM will output pulses
48
49    unsigned long DistanceMeasured=pulseIn(URPWM,LOW);
50
51    if(DistanceMeasured==50000){             // the reading is invalid.
52      Serial.print("Invalid");
53   }
54    else{
55      Distance=DistanceMeasured/50;          // every 50us low level stands for 1cm
56   }
57  Serial.print("Distance=");
58  Serial.print(Distance);
59  Serial.println("cm");
60}
```

Serial control protocol

Serial setting：Port rate: 9600; Parity: none; Stop bit: 1

Command: Control command consists of four bits, command+data0+data1+sum. Sum=Low 8 bit of the sum of command+data0+data1.

Command Format	Function	Description
0x11＋NC＋NC＋Sum (Sample: 0x11 0x00 0x00 0x11)	**Enable 16 bit temperature reading**	**Reading the temperature, the return data format will be:** **0x11+High(temperature)+Low(temperature)+SUM** **If the temperature is above 0, the first four bits of High will be all 0. If the temperature is below 0, the first four bits of High will be all 1.**

		The last 4 bits of High together with the Low bits stands for 12bits temperature. The resolution is 0.1. When the reading is invalid, it returns 0x11+0xFF+0xFF+SUM
0x22＋Degree＋NC＋SUM (Sample: 0x22 0x00 0x00 0x22)	**Enable 16 bit distance reading**	**The degree in the command is used to control a servo motor to rotate corresponding degree.** **Degree: 0-46 stands for 0-270 degrees, for example, 3 stands for 18 degrees.** **Return data format will be: 0x22＋High(distance)＋Low(distance)+SUM. When the reading is invalid, it returns 0x22+0xFF+0xFF+SUM**
0x33＋Add＋NC＋SUM	**Enable internal EEPROM reading**	**Return data will be 0x33＋Add＋Data＋SUM.**
0x44＋Add＋Data＋SUM (Sample: 0x44 0x02 0xbb 0x01) Enable PWM mode	**Enable internal EEPROM writing**	**Written data can only from 0-255.** **Address 0x00-0x02 is used to configure the mode. 0x00 – threshold distance (Low) 0x01 – threshold distance (High) 0x02 – Operation Mode (0xaa for autonomous mode) (0xbb for PWM passive control mode)** **The return data format will be: 0x44＋Add＋Data＋SUM**

Note:NC stands for any data，SUM stands for sum, Add stands for address.

1．PWN_ON must be set to High to enable sensor.

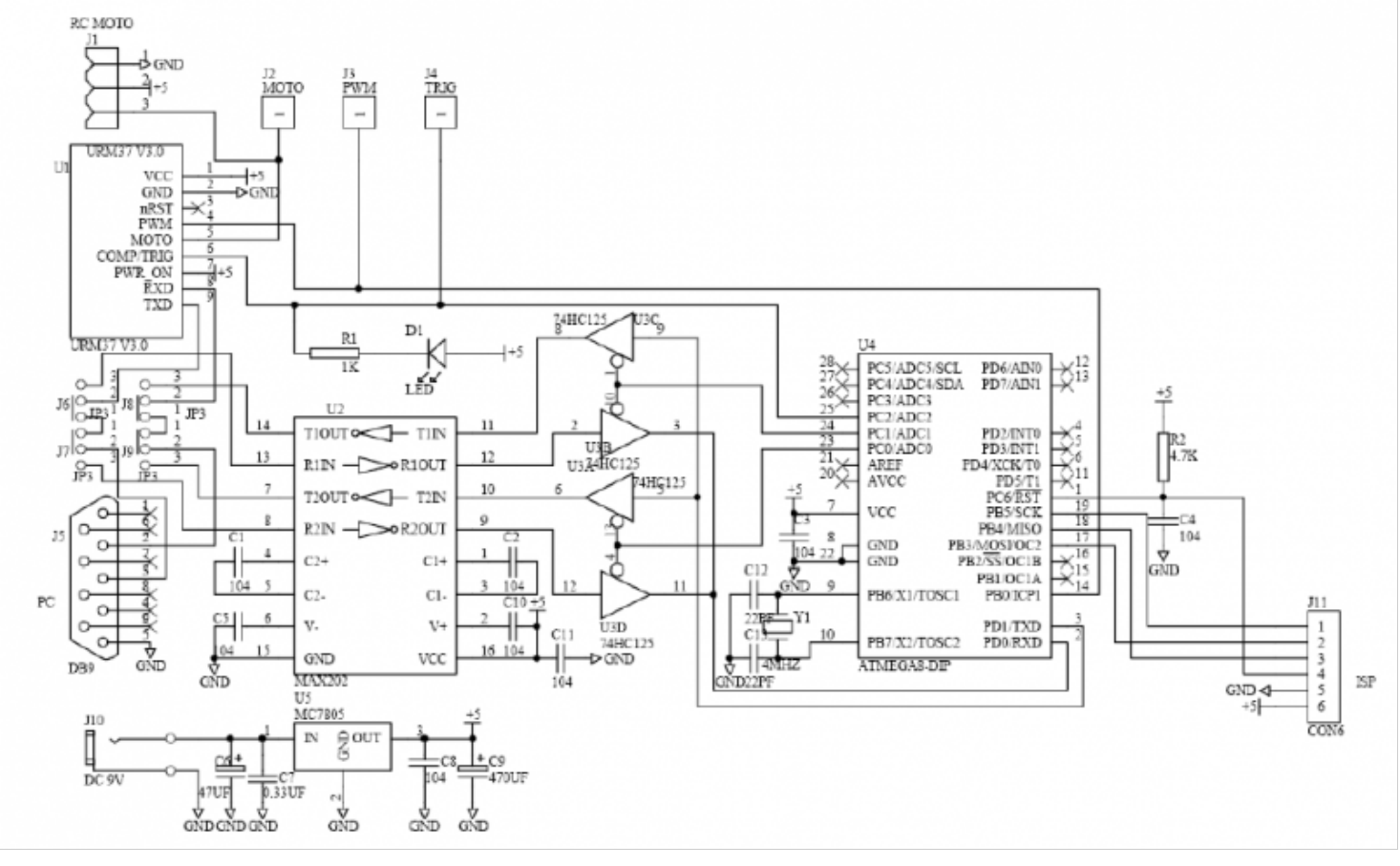
Examples: Function to calculate the temperature:

```
?
1IF(HightByte>=0xF0)
2{
3Temperature= ((HightByte-0xF0)*256-LowByte)/10
4 }
5Else
6{
7Temperature= ((HightByte)*256-LowByte)/10
8}
```

Servo control command reference table:

DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HEX	0	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Degree	0	6	12	18	24	29	35	41	47	53	59	65	70	76	82	88
DEC	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
HEX	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Degree	94	100	106	112	117	123	129	135	141	147	153	159	164	170	176	182
DEC	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	

HEX	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	
Degree	188	194	200	206	211	217	223	229	235	241	247	252	258	264	270	



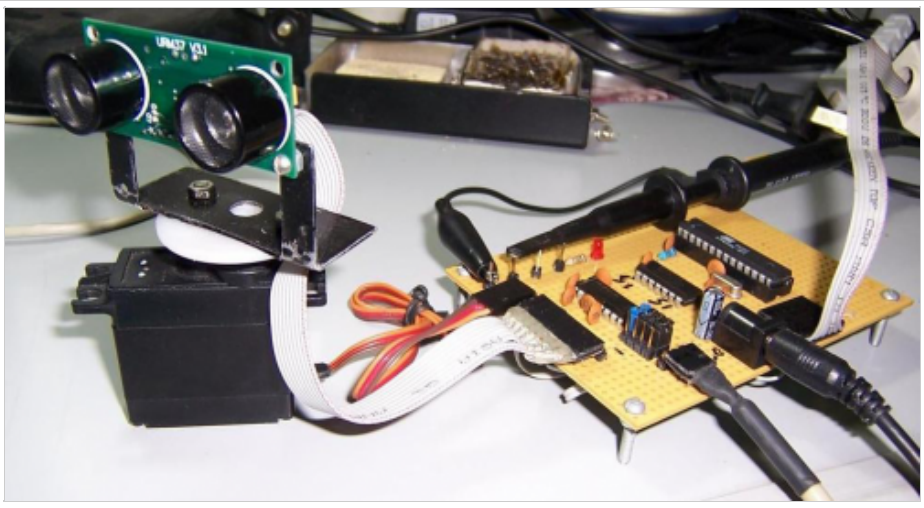


Figure UMRV3.2 control a servo provides 270 degree scanning area

V3.2 Help Mate Download:

Arduino Sketch

Use a servo to control the position,and the ultrasonic sensor to judge the distance.Here is the sketch.

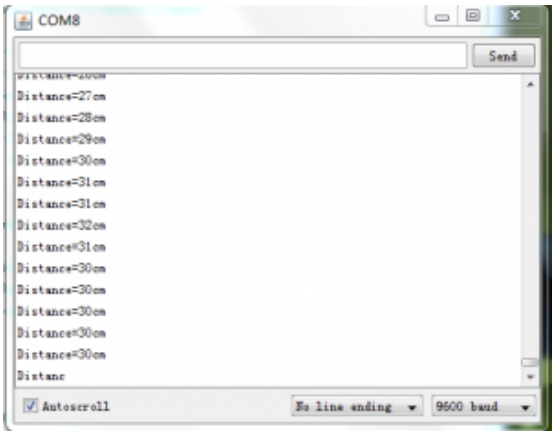
NOTE: Please put the sensor jumpers to TTL mode. See above for a picture indicating TTL mode

?

```
1
2  // # Editor    : Jiang from DFRobot
3  // # Data      : 24.07.2012
4
5  // # Product name:ultrasonic scanner Kit
6  // # Product SKU:SEN0001
7  // # Version :  0.2
8
9  // # Description:
10 // # The Sketch for scanning 180 degree area 4-500cm detecting range
11
12 // # Connection:
13 // #       Pin 1 VCC (URM V3.2) -> VCC (Arduino)
14 // #       Pin 2 GND (URM V3.2) -> GND (Arduino)
15 // #       Pin 4 PWM (URM V3.2) -> Pin 3 (Arduino)
16 // #       Pin 6 COMP/TRIG (URM V3.2) -> Pin 5 (Arduino)
17 // # Pin mode: PWM
18 // # Working Mode: PWM passive control mode.
19 // # If it is your first time to use it,please make sure the two jumpers to the right hand
20 // # side of the device are set to TTL mode. You'll also find a secondary jumper on
21 // # the left hand side, you must break this connection or you may damage your device.
22
23 #include <Servo.h>                                    // Include Servo library
24 Servo myservo;                                        // create servo object to control a servo
25
26 int pos=0;                                           // variable to store the servo position
   int URPWM=3;                                         // PWM Output 0-25000us,every 50us represent 1cm
```

```
int URTRIG=5;                                   // PWM trigger pin
boolean up=true;                                // create a boolean variable
unsigned long time;                             // create a time variable
unsigned long urmTimer = 0;                     // timer for managing the sensor reading flash rate

unsigned int Distance=0;
uint8_t EnPwmCmd[4]={0x44,0x22,0xbb,0x01};      // distance measure command

void setup(){                                   // Serial initialization
  Serial.begin(9600);                           // Sets the baud rate to 9600
  myservo.attach(9);                            // Pin 9 to control servo
  PWM_Mode_Setup();
}

void loop(){
   if(millis()-time>=20){                       // interval 0.02 seconds
     time=millis();                             // get the current time of programme
     if(up){                                    // judge the condition
      if(pos>=0 && pos<=179){
        pos=pos+1;                              // in steps of 1 degree
        myservo.write(pos);                     // tell servo to go to position in variable 'pos'
      }
      if(pos>179)  up= false;                   // assign the variable again
    }
     else {
      if(pos>=1 && pos<=180){
        pos=pos-1;
        myservo.write(pos);
      }
      if(pos<1)  up=true;
    }
   }

   if(millis()-urmTimer>50){
     urmTimer=millis();
     PWM_Mode();
   }
 }

 void PWM_Mode_Setup(){
  pinMode(URTRIG,OUTPUT);                       // A low pull on pin COMP/TRIG
  digitalWrite(URTRIG,HIGH);                    // Set to HIGH

  pinMode(URPWM, INPUT);                        // Sending Enable PWM mode command

  for(int i=0;i<4;i++){
      Serial.write(EnPwmCmd[i]);
   }
}

void PWM_Mode(){                                // a low pull on pin COMP/TRIG  triggering a sensor
reading
    digitalWrite(URTRIG, LOW);
    digitalWrite(URTRIG, HIGH);                 // reading Pin PWM will output pulses

    unsigned long DistanceMeasured=pulseIn(URPWM,LOW);

    if(DistanceMeasured==50000){                // the reading is invalid.
      Serial.print("Invalid");
```

```
  }
   else{
     Distance=DistanceMeasured/50;                 // every 50us low level stands for 1cm
  }
 Serial.print("Distance=");
 Serial.print(Distance);
 Serial.println("cm");
}
```



Resources

- **Arduino Library from milesburton(IDE 0023 and below) (http://milesburton.com/URM37_Ultrasonic_Distance_Measurement_Library)**
- **Arduino Library from Lauren(Only Arduino IDE 1.0) (http://www.dfrobot.com/image/data/SEN0001/URM37%20library%20for%20Arduino%20IDE%201.0.rar)**

Retrieved from "http://www.dfrobot.com/wiki/index.php?title=URM37_V3.2_Ultrasonic_Sensor_(SKU:SEN0001)&oldid=27098"
Categories: Product Manual | SEN Series

- This page was last modified on 12 August 2014, at 04:13.
- This page has been accessed 63,928 times.