URM37 V3.2 Ultrasonic Sensor (SKU: SEN0001)

Contents

- 1 Introduction
- 2 Specification
  - 2.1 Compare with other ultrasonic sensor
- 3 Hardware requirements
- 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.

Specification

- Power: +5V
- Current: <20mA
- Working temperature: -10°C ~ +70°C
- 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
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 requirements

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](image1)

![TTL Mode](image2)

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.
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
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>

\[
\text{checkSum} = \text{Low 8 bit of the sum of command+data0+data1}
\]

```c
int cmd1[]={
  0x44,0x00,0x10,0x54}; //low byte stored in the sensor for the distance threshold.
int cmd2[]={
  0x44,0x01,0x00,0x45}; //high byte, write 0x0010 into address 0x01 and 0x00, so the threshold is set to 16cm
int cmd3[]={
  0x44,0x02,0xaa,0xf0}; // Autonomous mode. write 0xaa into address 0x02
```

23// int cmmd3[]={
24// 0x44,0x02,0xbb,0x01}; // PWM mode. write 0xbb into address 0x02
25int i;
26
27void setup(){
28 Serial.begin(9600);  // Sets the baud rate to 9600
29 A_Mode_Setup();     // PWM mode setup function
30}
31
32void loop();
33{
34}
35
36void A_Mode_Setup(){
37 //write the data into the URM37 EEPROM
38 for (i=0;i<4;i++)
39    Serial.write(cmmd3[i]);
40    delay(200);
41
42 for (i=0;i<4;i++)
43    Serial.write(cmmd1[i]);
44    delay(200);
45
46 for (i=0;i<4;i++)
47    Serial.write(cmmd2[i]);
48    delay(200);
49
50}

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.
5  pinMode(pin, OUTPUT);
6  attachInterrupt(0, user_diy, CHANGE);  //The ON/OFF switch can be used as a signal of interruption
7 }
8
9 void loop(){
10  digitalWrite(pin, state);
11}
12
13 void user_diy()  //user can give your own code in this interrupt function
14{
15  state = !state;
16}

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

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");
  }
}

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.

<table>
<thead>
<tr>
<th>Command Format</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x11 + NC + NC + Sum</td>
<td>Enable 16 bit temperature</td>
<td>Reading the temperature, the return data format will be:</td>
</tr>
<tr>
<td>(Sample: 0x11 0x00 0x00 0x11)</td>
<td>reading</td>
<td>0x11+High(temperature)+Low(temperature)+SUM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
The last 4 bits of High together with the Low bits stands for 12 bits temperature. The resolution is 0.1. When the reading is invalid, it returns 0x11+0xFF+0xFF+SUM

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

Enable 16 bit distance reading

0x22 + Degree + NC + SUM (Sample: 0x22 0x00 0x00 0x22 )

Enable internal EEPROM reading

0x33 + Add + NC + SUM

Enable internal EEPROM writing

0x44 + Add + Data + SUM (Sample: 0x44 0x02 0xbb 0x01) Enable PWM mode

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:

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

Servo control command reference table:

<table>
<thead>
<tr>
<th>DEC</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>0</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>0A</td>
<td>0B</td>
<td>0C</td>
<td>0D</td>
<td>0E</td>
<td>0F</td>
</tr>
<tr>
<td>Degree</td>
<td>0</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>29</td>
<td>35</td>
<td>41</td>
<td>47</td>
<td>53</td>
<td>59</td>
<td>65</td>
<td>70</td>
<td>76</td>
<td>82</td>
<td>88</td>
</tr>
<tr>
<td>DEC</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>HEX</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>1A</td>
<td>1B</td>
<td>1C</td>
<td>1D</td>
<td>1E</td>
<td>1F</td>
</tr>
<tr>
<td>Degree</td>
<td>94</td>
<td>100</td>
<td>106</td>
<td>112</td>
<td>117</td>
<td>123</td>
<td>129</td>
<td>135</td>
<td>141</td>
<td>147</td>
<td>153</td>
<td>159</td>
<td>164</td>
<td>170</td>
<td>176</td>
<td>182</td>
</tr>
<tr>
<td>DEC</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>HEX</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>2A</td>
<td>2B</td>
<td>2C</td>
<td>2D</td>
<td>2E</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Degree</td>
<td>188</td>
<td>194</td>
<td>200</td>
<td>206</td>
<td>211</td>
<td>217</td>
<td>223</td>
<td>229</td>
<td>235</td>
<td>241</td>
<td>247</td>
<td>252</td>
<td>258</td>
<td>264</td>
<td>270</td>
<td></td>
</tr>
</tbody>
</table>

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.

```cpp
#include <Servo.h> // Include Servo library
Servo myservo; // create servo object to control a servo

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 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(){ // interval 0.02 seconds
  if(millis()-time>=20){ // get the current time of programme
    time=millis(); // judge the condition
    if(up){
      if(pos>=0 & pos<=179){
        pos=pos+1;
        myservo.write(pos);
      }
      if(pos>179) up=false;
    }
    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
    digitalWrite(URTRIG, LOW); // reading Pin PWM will output pulses
    digitalWrite(URTRIG, HIGH);
    unsigned long DistanceMeasured=pulseIn(URPWM,LOW);
    if(DistanceMeasured==50000){ // the reading is invalid.
      Serial.print("Invalid");
    }
  }
```c
86 } 
87 else{
88     Distance=DistanceMeasured/50;  // every 50us low level stands for 1cm 
89 } 
90 Serial.print("Distance=");
91 Serial.print(Distance);
92 Serial.println("cm");
93}
```

**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)


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.