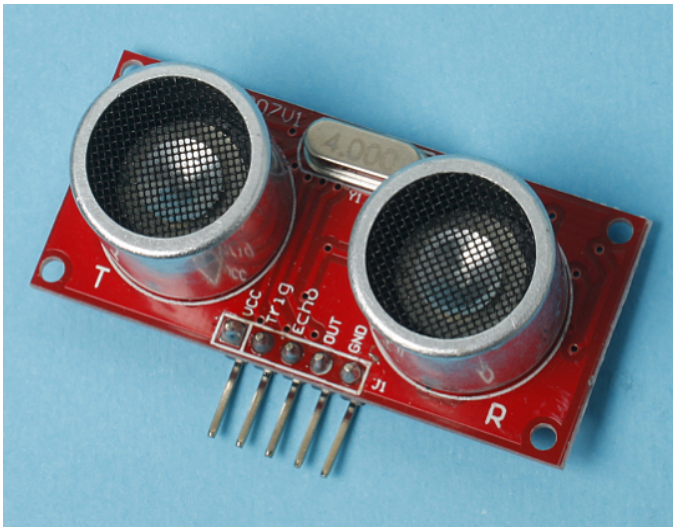
40Hz Ultrasonic Range Detection Sensor

Model:DYP-ME007V1



Module Working Principle:

(1) Adopt IO trigger through supplying at least 10us sequence of high level signal,

(2) The module automatically send eight 40khz square wave and automatically detect whether receive the returning pulse signal,

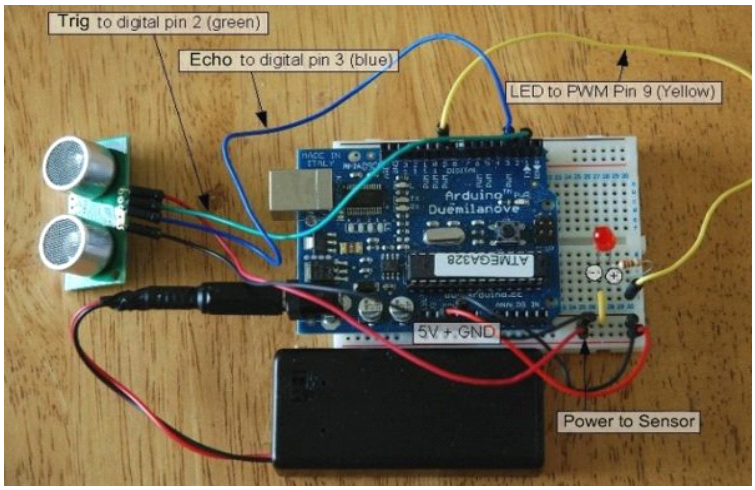
(3) If there is signals returning, through outputting high level

and the time of high level continuing is the time of that from the ultrasonic transmitting to receiving.

Test distance = (high level time * sound velocity (340M/S) / 2,

The circuit:

Very, very simple circuit, I've used the breadboard to share the GND connection and to add the LED which I could probably have done with out the breadboard. You'll see the most complex thing is the code later on.



The sketch:

All the work is done here, I've added code that averages the distance readings to remove some of the jitter in the results as the DYP-ME007 is calculating distances very rapidly and there can be a lot of fluctuation. Also I convert the time in microseconds to distance by dividing the time by 58.
Why 58? Well because if you take the time in microseconds for a pulse to be sent and received e.g. for 1 meter it takes about 5764 microseconds – at least from my wall anyway. If I divide this time by the distance in cm in I will get 57.64 so I just round this up – you can calculate distance in any other unit with this method.
Here I've also decided that for every cm under 255 my LED will get 1 step brighter. I've been lazy here for the sake of the sensors 3 metre range I didn't see the point in making this any more complicated. Otherwise I would calculate the brightness on the percentile of proximity out of total range.

Official test Code Example - 1:

```
 #include <c8051f020.h>

#define uchar unsigned char
#define uint    unsigned int

sbit P0_0=P0^0;
sbit P0_1=P0^1;
bit    flag;

void delay(uint z)
{
      uint x,y;
      for(x=z;x>0;x--)
      for(y=0;y<110;y++);
}

void T0T1_Init()
{
      CKCON=0x00;
      TMOD=0x11;
      TH0=(65535-25)/256;
      TL0=(65535-25)%256;
      TH1=0;
      TL1=0;
      TR0=0;
      TR1=0;
      ET0=1;
      ET1=1;
      EA=1;
}
```

```
void main()
{
      WDTCN = 0xde;
      WDTCN = 0xad;
//    OSCXCN=0x67;
//    delay(100);
  //  OSCICN=0x08;
      P0MDOUT=0x00;
      flag=0;
      P0_1=0;
      P0_0=1;



      T0T1_Init();

//    P1MDOUT=0xff;
//    P3MDOUT=0XFF;
      P74OUT=0xff;
      P4=0x01;


      XBR2=0x40;
      XBR1=0x04;
//    duan=0;
//   wei=0;

      P0_1=1;
      TR0=1;
      while(!flag);
      TR0=0;
      P0_1=0;
      flag=0;
      while(!P0_0);
      TR1=1;
      IT0=1;
      EX0=1;
      while(1)
       {
      }
}
```

```
void INT0_ISR() interrupt 0
{     //EA=0;
      TR1=0;
      //delay(8000);
      P4=(P4<<1|P4>>(8-1));
      //TH1=0;
//    TL1=0;
//    a=(TH1*256+TL1)*0.017;
//    bai=
/*    for(j=0;j<6;j++)
   {
      wei=1;
      P1=table1[j];
      wei=0;

      duan=1;
      P1=table[j];
      duan=0;
    delay(2000);
   }*/
}

void T0_ISR() interrupt 1
{
      TH0=(65535-25)/256;
      TL0=(65535-25)%256;
      flag=1;
}
```

Official test Code Example - 2:

```
// variables to take x number of readings and then average them
// to remove the jitter/noise from the DYP-ME007 sonar readings
const int numOfReadings = 10;                    // number of readings to take/ items in the
array
int readings[numOfReadings];                     // stores the distance readings in an array
int arrayIndex = 0;                              // arrayIndex of the current item in the
array
int total = 0;                                   // stores the cumlative total
int averageDistance = 0;                         // stores the average value
// setup pins and variables for DYP-ME007 sonar device
int echoPin = 2;                                 // DYP-ME007 echo pin (digital 2)
int initPin = 3;                                 // DYP-ME007 trigger pin (digital 3)
unsigned long pulseTime = 0;                     // stores the pulse in Micro Seconds
unsigned long distance = 0;                      // variable for storing the distance (cm)
// setup pins/values for LED
int redLEDPin = 9;                               // Red LED, connected to digital PWM
pin 9
int redLEDValue = 0;                             // stores the value of brightness for the
LED (0 = fully off, 255 = fully on)
```

```
//setup
void setup() {
  pinMode(redLEDPin, OUTPUT);                    // sets pin 9 as output
  pinMode(initPin, OUTPUT);                      // set init pin 3 as output
  pinMode(echoPin, INPUT);                       // set echo pin 2 as input
  // create array loop to iterate over every item in the array
  for (int thisReading = 0; thisReading < numOfReadings; thisReading++) {
readings[thisReading] = 0;
 }
// initialize the serial port, lets you view the
 // distances being pinged if connected to computer
       Serial.begin(9600);
 }
// execute
void loop() {
digitalWrite(initPin, HIGH);                    // send 10 microsecond pulse
delayMicroseconds(10);                          // wait 10 microseconds before turning off
digitalWrite(initPin, LOW);                     // stop sending the pulse
pulseTime = pulseIn(echoPin, HIGH);             // Look for a return pulse, it should be high
as the pulse goes low-high-low
distance = pulseTime/58;                        // Distance = pulse time / 58 to convert to
cm.
 total= total - readings[arrayIndex];           // subtract the last distance
readings[arrayIndex] = distance;                // add distance reading to array
total= total + readings[arrayIndex];            // add the reading to the total
arrayIndex = arrayIndex + 1;                    // go to the next item in the array
// At the end of the array (10 items) then start again
if (arrayIndex >= numOfReadings)    {
      arrayIndex = 0;
  }
  averageDistance = total / numOfReadings;      // calculate the average distance
  // if the distance is less than 255cm then change the brightness of the LED
  if (averageDistance < 255) {
      redLEDValue = 255 - averageDistance;      // this means the smaller the distance the
brighterthe LED.
  }
  analogWrite(redLEDPin, redLEDValue);          // Write current value to LED pins
  Serial.println(averageDistance, DEC);         // print out the average distance to the
debugger
  delay(100);                                   // wait 100 milli seconds before looping
again
}
```

Official test Code Example - 3:

```
////////////////////////////////////////////////////////////////////////////
//
//        PIC16F877 + DYP-ME007 + LCD03 example
//        Written October 2005 by Gerald Coe, using HITECH PIC16 compiler
//
//            Note - assumes a 20MHz crystal, which is 5MHz timer clock
//            A 1:4 prescaler is used to give a 1.25MHz timer count (0.8uS per tick)
//
//        This code is Freeware - Use it for any purpose you like.
//
////////////////////////////////////////////////////////////////////////////


#include <pic.h>
#include <stdio.h>

__CONFIG(0x3b32);

#define trig    RB0
#define echo    RB1

void clrscn(void);                                  // prototypes
void cursor(char pos);
void print(char *p);
void setup(void);
unsigned int get_srf04(void);

char s[21];                                         // buffer used to hold text to print

void main(void)
{
unsigned int range;
    setup();                                        // sets up the PIC16F877 I2C port
    clrscn();                                       // clears the LCD03 disply
    cursor(2);                                      // sets cursor to 1st row of LCD03
    sprintf(s,"SRF04 Ranger Test");         // text, printed into our buffer
    print(s);                                       // send it to the LCD03

    while(1) {                                      // loop forever
        range = get_srf04();                        // get range from srf04 (round trip flight
time in 0.8uS units)
        cursor(24);                                 // sets cursor to 2nd row of LCD03
        sprintf(s,"Range = %dcm    ", range/72);    // convert to cm
        print(s);                                   // send it to the LCD03
        cursor(44);                                 // sets cursor to 3rd row of LCD03
        sprintf(s,"Range = %dinch    ", range/185); // convert to inches
        print(s);                                   // send it to the LCD03
```

```
        TMR1H = 0;                                  // 52mS delay - this is so that the
SRF04 ranging is not too rapid
        TMR1L = 0;                                  // and the previous pulse has faded
away before we start the next one
        T1CON = 0x21;                               // 1:4 prescale and running
        TMR1IF = 0;
        while(!TMR1IF);                             // wait for delay time
        TMR1ON = 0;                                 // stop timer
    }
}

unsigned int get_srf04(void)
{
    TMR1H = 0xff;                                   // prepare timer for 10uS pulse
    TMR1L = -14;
    T1CON = 0x21;                                   // 1:4 prescale and running
    TMR1IF = 0;
    trig = 1;                                   // start trigger pulse
    while(!TMR1IF);                             // wait 10uS
    trig = 0;                                   // end trigger pulse
    TMR1ON = 0;                                 // stop timer

    TMR1H = 0;                                  // prepare timer to measure echo pulse
    TMR1L = 0;
    T1CON = 0x20;                               // 1:4 prescale but not running yet
    TMR1IF = 0;
    while(!echo && !TMR1IF);            // wait for echo pulse to start (go high)
    TMR1ON = 1;                                 // start timer to measure pulse
    while(echo && !TMR1IF);             // wait for echo pulse to stop (go low)
    TMR1ON = 0;                                 // stop timer
    return (TMR1H<<8)+TMR1L;            // TMR1H:TMR1L contains flight time of the pulse
in 0.8uS units
}
void clrscn(void)
{
    SEN = 1;                                    // send start bit
    while(SEN);                                 // and wait for it to clear

    SSPIF = 0;
    SSPBUF = 0xc6;                              // LCD02 I2C address
    while(!SSPIF);                              // wait for interrupt
    SSPIF = 0;                                  // then clear it.

    SSPBUF = 0;                                 // address of register to write to
    while(!SSPIF);                              //
    SSPIF = 0;                                  //

    SSPBUF = 12;                            // clear screen
    while(!SSPIF);                              //
    SSPIF = 0;                                  //
```

```
    SSPBUF = 4;                 // cursor off
    while(!SSPIF);              //
    SSPIF = 0;                  //

    PEN = 1;                    // send stop bit
    while(PEN);                 //
}

void cursor(char pos)
{
    SEN = 1;                    // send start bit
    while(SEN);                 // and wait for it to clear

    SSPIF = 0;
    SSPBUF = 0xc6;              // LCD02 I2C address
    while(!SSPIF);              // wait for interrupt
    SSPIF = 0;                  // then clear it.

    SSPBUF = 0;                 // address of register to write to
    while(!SSPIF);              //
    SSPIF = 0;                  //

    SSPBUF = 2;                 // set cursor
    while(!SSPIF);              //
    SSPIF = 0;                  //
    SSPBUF = pos;               //
    while(!SSPIF);              //
    SSPIF = 0;                  //

    PEN = 1;                    // send stop bit
    while(PEN);                 //
}

void print(char *p)
{
    SEN = 1;                    // send start bit
    while(SEN);                 // and wait for it to clear

    SSPIF = 0;
    SSPBUF = 0xc6;              // LCD02 I2C address
    while(!SSPIF);              // wait for interrupt
    SSPIF = 0;                  // then clear it.

    SSPBUF = 0;                 // address of register to write to
    while(!SSPIF);              //
    SSPIF = 0;                  //
```

```
	while(*p) {
		SSPBUF = *p++;		// write the data
		while(!SSPIF);		//
		SSPIF = 0;		//
	}

	PEN = 1;			// send stop bit
	while(PEN);			//
}


void setup(void)
{
unsigned long x;
	TRISB = 0xfe;			// RB0 (trig) is output
	PORTB = 0xfe;			// and starts low

	TRISC = 0xff;
	PORTC = 0xff;

	SSPSTAT = 0x80;
	SSPCON = 0x38;
	SSPCON2 = 0x00;
	SSPADD = 50;			// SCL = 91khz with 20Mhz Osc

	for(x=0; x<300000L; x++);	// wait for LCD03 to initialise
}
```