Jameco Part Number 282861

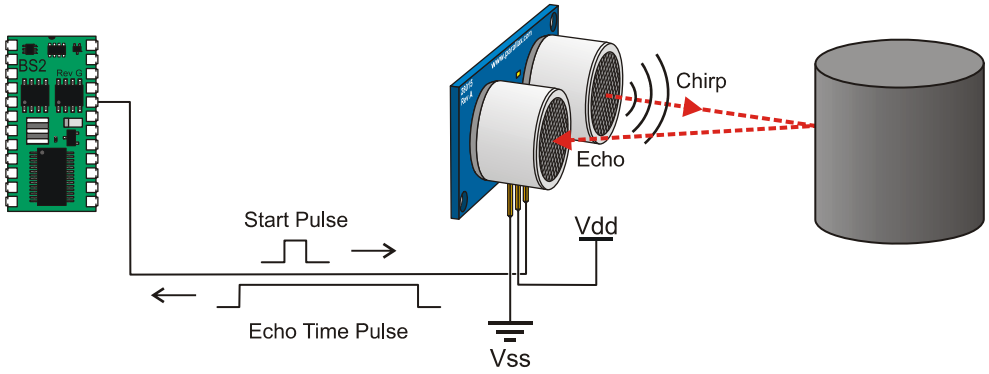
Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

PING)))™ Ultrasonic Distance Sensor (#28015)

The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, SX or Propeller chip, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.



Features

- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2 µs min, 5 µs typ.
- Echo pulse: positive TTL pulse, 115 µs minimum to 18.5 ms maximum.
- RoHS Compliant

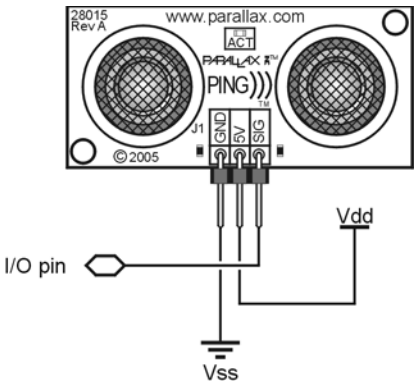
Key Specifications

- Supply voltage: +5 VDC
- Supply current: 30 mA typ; 35 mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1" spacing (ground, power, signal)
- Operating temperature: 0 – 70° C.
- Size: 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

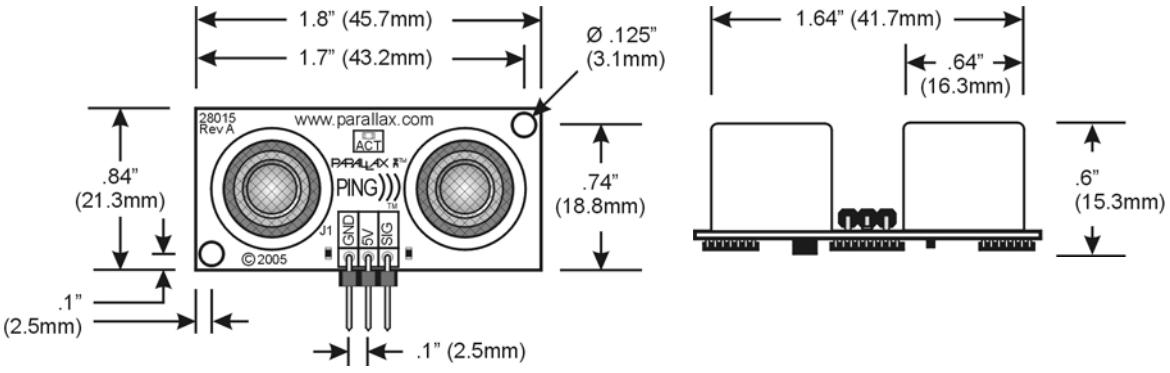
Pin Definitions

GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged into a directly into solderless breadboard, or into a standard 3-wire extension cable (Parallax part #805-000012).

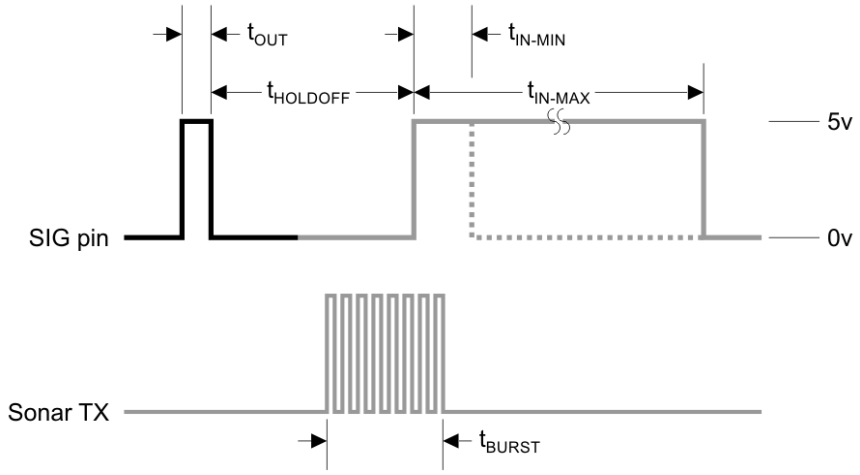


Dimensions



Communication Protocol

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

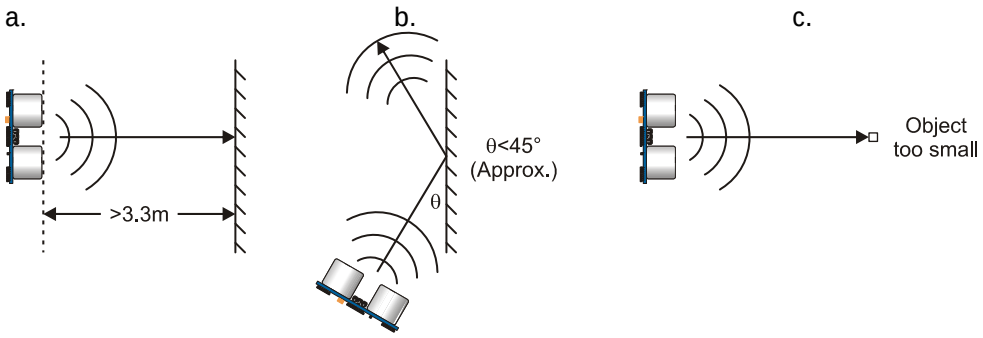


▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

Practical Considerations for Use

Object Positioning

The PING))) sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING))) sensor is mounted low on your device, you may detect sound reflecting off of the floor.



Target Object Material

In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING))) sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device. See the "Water Level with PING)))" document on the 28015 product page at www.parallax.com for more information.

Air Temperature

Temperature has an effect on the speed of sound in air that is measurable by the PING))) sensor. If the temperature (°C) is known, the formula is:

$$C_{air} = 331.5 + (0.6 \times T_C)\ m/s$$

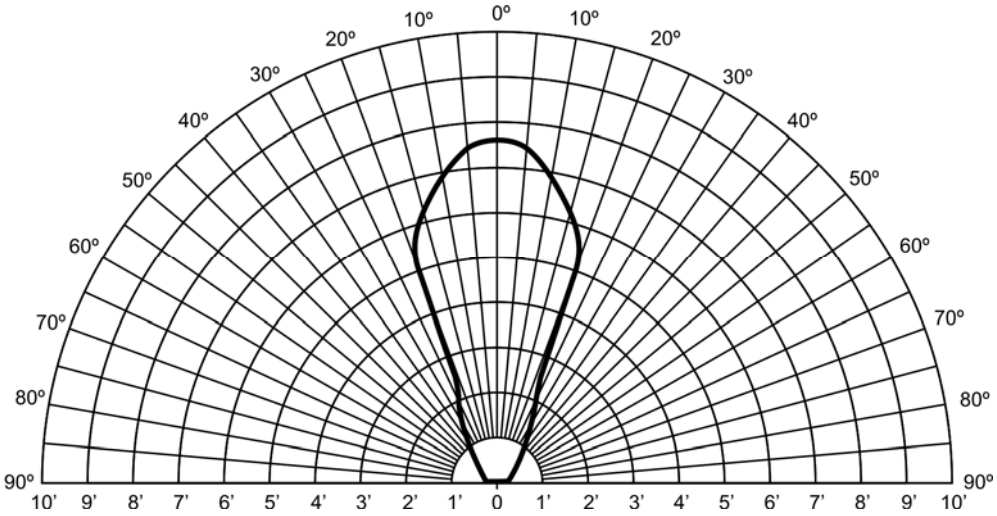
The percent error over the sensor's operating range of 0 to 70 ° C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the example BS2 program given in the Example Programs section below). Percent error and conversion constant calculations are introduced in Chapter 2 of *Smart Sensors and Applications*, a Stamps in Class text available for download from the 28029 product page at www.parallax.com.

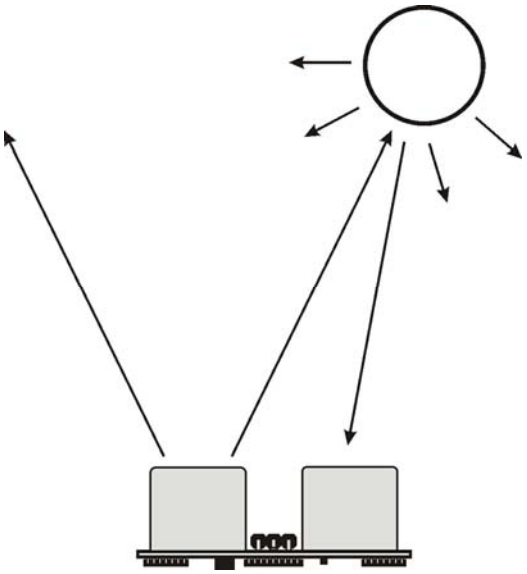
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

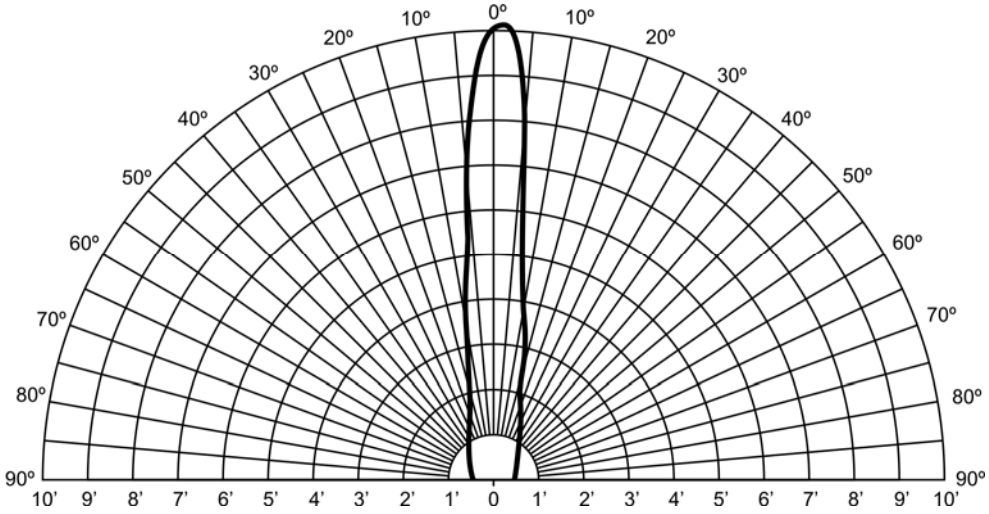
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

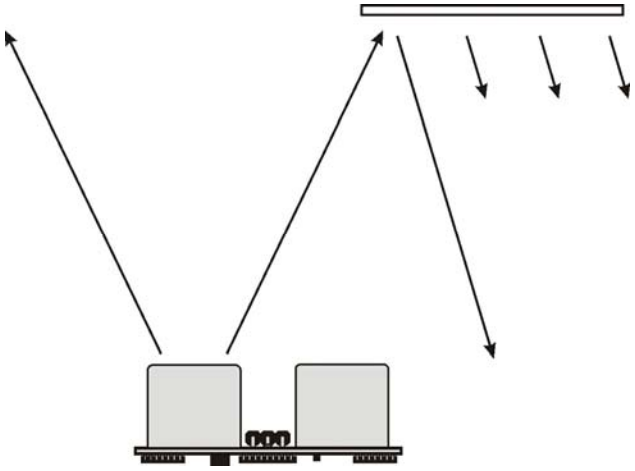




Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
Target positioned parallel to backplane of sensor

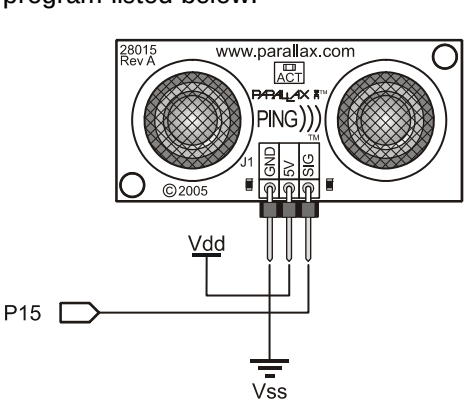


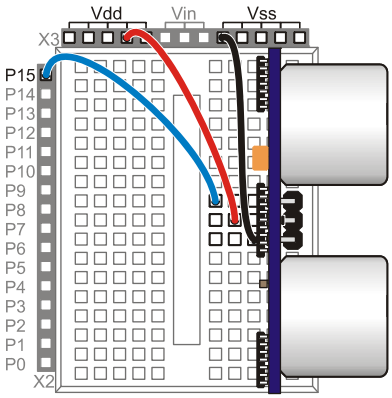


Example Programs and Applications

BASIC Stamp 2

This circuit allows you to quickly connect your PING))) sensor to a BASIC Stamp® 2 via the Board of Education® breadboard area. The PING))) module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit will work with the example BASIC Stamp program listed below.

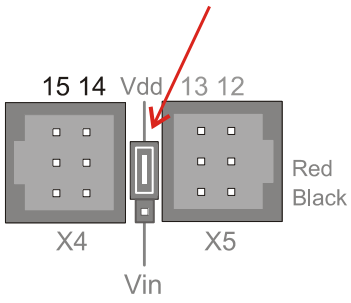




Extension Cable and Port Cautions for the Board of Education

If you are connecting your PING))) sensor to a Board of Education platform using an extension cable, follow these steps:

1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown. Then plug the cable into the port, matching the wire color to the labels next to the port.
4. If your Board of Education servo ports do not have a jumper, **do not use them with the PING))) sensor**. These ports only provide Vin, not Vdd, and this may damage your PING))) sensor. Go to the next step.
5. Connect the cable directly to the breadboard with a 3-pin header as shown above. Then, use jumper wires to connect Black to Vss, Red to Vdd, and White to I/O pin P15.



Board of Education Servo Port Jumper, Set to Vdd

Example Program: PingMeasureCmAndIn.bs2

This example BS2 program is an excerpt from Chapter 2 of the Stamps in Class text *Smart Sensors and Applications*. Additional PBASIC programs, one for the BS1 and another than runs on any model of BASIC Stamp 2 (BS2, BS2e, BS2sx, BS2p, BS2pe, BS2px) can be downloaded from the 28015 product page.

```
' Smart Sensors and Applications - PingMeasureCmAndIn.bs2
' Measure distance with Ping))) sensor and display in both in & cm

' {$STAMP BS2}
' {$PBASIC 2.5}

' Conversion constants for room temperature measurements.
CmConstant    CON   2260
InConstant    CON   890

cmDistance  VAR   Word
inDistance  VAR   Word
time        VAR   Word

DO

  PULSOUT 15, 5
  PULSIN 15, 1, time

  cmDistance = cmConstant ** time
  inDistance = inConstant ** time

  DEBUG HOME, DEC3 cmDistance, " cm"
  DEBUG CR, DEC3 inDistance, " in"

  PAUSE 100

LOOP
```

Propeller Microcontroller

```
{{
*************************************
*       Ping))) Object V1.1         *
*      (C) 2006 Parallax, Inc.      *
* Author:  Chris Savage & Jeff Martin *
* Started: 05-08-2006               *
*************************************

Interface to Ping))) sensor and measure its ultrasonic travel time.  Measurements can be in
units of time or distance.  Each method requires one parameter, Pin, that is the I/O pin that
is connected to the Ping)))'s signal line.

   ┌──────────────────────┐
   │ ┌──┐ PING))) ┌──┐    │     Connection To Propeller
   │ └──┘         └──┘    │     Remember PING))) Requires
   │    GND +5V SIG       │     +5V Power Supply
   └──────────────────────┘
          │   ┌┐   ≷ 1K
          ▽   └┘    └─ Pin

-------------------------REVISION HISTORY-------------------------
 v1.1 - Updated 03/20/2007 to change SIG resistor from 10K to 1K
}}

CON
  TO_IN = 73_746                                                ' Inches
  TO_CM = 29_034                                                ' Centimeters

PUB Ticks(Pin) : Microseconds | cnt1, cnt2
''Return Ping)))'s one-way ultrasonic travel time in microseconds

  outa[Pin]~                                                    ' Clear I/O Pin
  dira[Pin]~~                                                   ' Make Pin Output
  outa[Pin]~~                                                   ' Set I/O Pin
  outa[Pin]~                                                    ' Clear I/O Pin (> 2 µs pulse)
  dira[Pin]~                                                    ' Make I/O Pin Input
  waitpne(0, |< Pin, 0)                                         ' Wait For Pin To Go HIGH
  cnt1 := cnt                                                   ' Store Current Counter Value
  waitpeq(0, |< Pin, 0)                                         ' Wait For Pin To Go LOW
  cnt2 := cnt                                                   ' Store New Counter Value
  Microseconds := (||(cnt1 - cnt2) / (clkfreq / 1_000_000)) >> 1    ' Return Time in µs

PUB Inches(Pin) : Distance
''Measure object distance in inches

  Distance := Ticks(Pin) * 1_000 / TO_IN                         ' Distance In Inches

PUB Centimeters(Pin) : Distance
''Measure object distance in centimeters

  Distance := Millimeters(Pin) / 10                              ' Distance In Centimeters

PUB Millimeters(Pin) : Distance
''Measure object distance in millimeters

  Distance := Ticks(Pin) * 10_000 / TO_CM                        ' Distance In Millimeters
```

The ping.spin object is used in an example project with the Parallax 4 x 20 Serial LCD (#27979) to display distance measurements. The complete Project Archive can be downloaded from the Propeller Object Exchange at http://obex.parallax.com.

```
_____________________________________
Parallax Propeller Chip Project Archive
_____________________________________

 Project :  "ping_demo"

Archived :  Tuesday, December 18, 2007 at 3:29:46 PM

    Tool :  Propeller Tool version 1.05.8


            ping_demo.spin
               |
               ├──Debug_Lcd.spin
               |     |
               |     ├──Serial_Lcd.spin
               |     |     |
               |     |     └──Simple_Serial.spin
               |     |
               |     └──Simple_Numbers.spin
               |
               └──ping.spin
```

Javelin Stamp Microcontroller

This class file implements several methods for using the PING))) sensor with the Javelin Stamp module.

```
package stamp.peripheral.sensor;

import stamp.core.*;


/**
 * This class provides an interface to the Parallax PING))) ultrasonic
 * range finder module.
 * <p>
 * <i>Usage:</i><br>
 * <code>
 *   Ping range = new Ping(CPU.pin0);       // trigger and echo on P0
 * </code>
 * <p>
 * Detailed documentation for the PING))) Sensor can be found at: <br>
 * http://www.parallax.com/detail.asp?product_id=28015
 * <p>
 *
 * @version 1.0 03 FEB 2005
 */
public final class Ping {

  private int ioPin;

  /**
   * Creates PING))) range finder object
   *
   * @param ioPin PING))) trigger and echo return pin
   */
  public Ping (int ioPin) {
    this.ioPin = ioPin;
  }


  /**
   * Returns raw distance value from the PING))) sensor.
   *
   * @return Raw distance value from PING)))
   */
  public int getRaw() {

    int echoRaw = 0;

    CPU.writePin(ioPin, false);              // setup for high-going pulse
    CPU.pulseOut(1, ioPin);                  // send trigger pulse
    echoRaw = CPU.pulseIn(2171, ioPin, true); // measure echo return

    // return echo pulse if in range; zero if out-of-range
    return (echoRaw < 2131) ? echoRaw : 0;
  }


  /*
   * The PING))) returns a pulse width of 73.746 uS per inch.  Since the
   * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
   * same as a one-way trip in 4.34 uS units.  Dividing 73.746 by 4.34 we
   * get a time-per-inch conversion factor of 16.9922 (x 0.058851).
   *
```

```
   * Values to derive conversion factors are selected to prevent roll-over
   * past the 15-bit positive values of Javelin Stamp integers.
   */

  /**
   * @return PING))) distance value in inches
   */
  public int getIn() {
    return (getRaw() * 3 / 51);              // raw * 0.058824
  }


  /**
   * @return PING))) distance value in tenths of inches
   */
  public int getIn10() {
    return (getRaw() * 3 / 5);               // raw / 1.6667
  }


  /*
   * The PING))) returns a pulse width of 29.033 uS per centimeter.  As the
   * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
   * same as a one-way trip in 4.34 uS units.  Dividing 29.033 by 4.34 we
   * get a time-per-centimeter conversion factor of 6.6896.
   *
   * Values to derive conversion factors are selected to prevent roll-over
   * past the 15-bit positive values of Javelin Stamp integers.
   */

  /**
   * @return PING))) distance value in centimeters
   */
  public int getCm() {
    return (getRaw() * 3 / 20);              // raw / 6.6667
  }


  /**
   * @return PING))) distance value in millimeters
   */
  public int getMm() {
    return (getRaw() * 3 / 2);               // raw / 0.6667
  }
}
```

This simple demo illustrates the use of the PING))) ultrasonic range finder class with the Javelin Stamp:

```
import stamp.core.*;
import stamp.peripheral.sensor.Ping;


public class testPing {

  public static final char HOME = 0x01;

  public static void main() {

    Ping range = new Ping(CPU.pin0);
    StringBuffer msg = new StringBuffer();

    int distance;
```

```
    while (true) {
      // measure distance to target in inches
      distance = range.getIn();

      // create and display measurement message
      msg.clear();
      msg.append(HOME);
      msg.append(distance);
      msg.append(" \"   \n");
      System.out.print(msg.toString());

      // wait 0.5 seconds between readings
      CPU.delay(5000);
    }
  }
}
```

Resources and Downloads

You can find additional resources for the PING))) sensor by searching the following product pages at www.parallax.com:

- Smart Sensors and Applications (a Stamps in Class text), #28029
- PING))) Mounting Bracket Kit – a servo-driven mount designed to attach to a Boe-Bot robot, #570-28015
- Extension cable with 3-in header, #805-00011 (10-in.) or #805-00012 (14-in.)

A video of a Boe-Bot robot using the PING))) sensor to scan its surroundings then drive to the closest object can be found under Resources > Video Library > Boe-Bot Robot Video Gallery.

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Water Level Measurement with the Ping))) Ultrasonic Distance Sensor (#28015)

General Description

Parallax customers frequently ask about measuring water level with the Ping))) Ultrasonic Distance Sensor. The Ping))) sensor isn't designed to be water resistant, and the specifications for the transducer don't provide any details for humidity sensitivity. Therefore, customers attempting to measure water level are doing so at their own risk of damaging their Ping))) hardware. At some point the humidity and moisture will likely damage your Ping))) Ultrasonic Distance Sensor, but depending on your circumstances and need this may or may not be a problem.

The purpose of this explanation is to convey the results of a brief test conducted at Parallax headquarters.

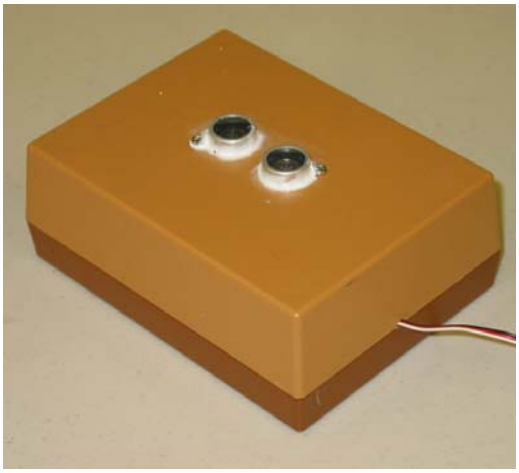
Ping))) Sensor Specifications

The Ping))) sensor's ultrasonic transducer emitter and detector have the following specifications:

Rated Freq	40 kHz
Sensitivity	-65 dB
Sound Pressure	115 dB
Capacitance	2 nF
Driving Voltage	0 VDC
Operating Temp	30° to 75 °C

No humidity exposure rating is provided by the transducer manufacturer.

Water Measurement Level Setup



We started our experiment by placing the Ping))) sensor in a two-inch diameter ABS plastic pipe. This provided readings to a maximum of five feet. Using a three-inch ABS pipe we obtained readings all the way to the end of the ten-foot length. This simple experiment showed that water level could be measured with a Ping))) sensor through the middle of an ABS pipe.

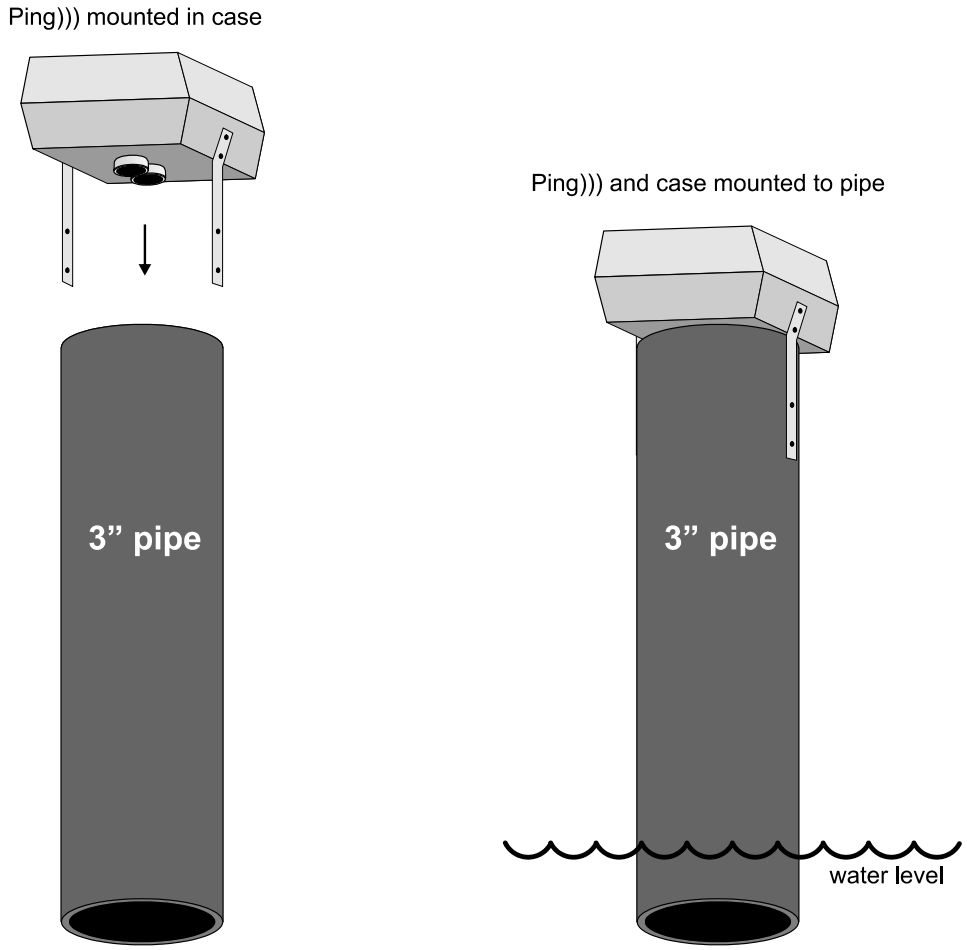
Next the Ping))) sensor was mounted in a plastic case to keep water away from most of the electronics. Silicon caulking was put around the transducers. The Ping)))

sensor setup was placed on top of the pipe with water in a bucket and left to rest for ten days to see if storage in a humid environment would be a problem. After this period, we tested the Ping))) sensor and it still operated (California is also a dry climate).

To continue with our tests, we attempted to measure the water level of boiling water (based on another customer request). We held the operating Ping))) sensor over boiling water for a few seconds at various heights, and it did indeed measure the distance to the boiling water. When steam was thick (closer to the water, and when the water was boiling hard) condensation droplets would accumulate on the transducers; we then moved it out of steam for a minute for the droplets to dry then continued with the test. The Ping sensor continued to operate throughout the 10-minute test. We concluded that the Ping))) sensor could sense the surface of the boiling water, but the measurements were affected when the cloud of steam was particularly thick.

The distance measurements through the ABS pipe were initially obtained using a ping pong ball taped to the end of a tape measure. The ball was moved up and down the pipe in one inch increments and the Ping))) showed accurate measurements.

Then the ten-foot pipe was inserted in a bucket of water and we moved the pipe up and down in the bucket. Though the bucket was only a foot deep the Ping))) showed accurate readings for each inch of movement ten feet away.



Chapter #1: Detect Distance with the Ping)))(TM) Ultrasonic Sensor

WHAT IS THE PING))) SENSOR?

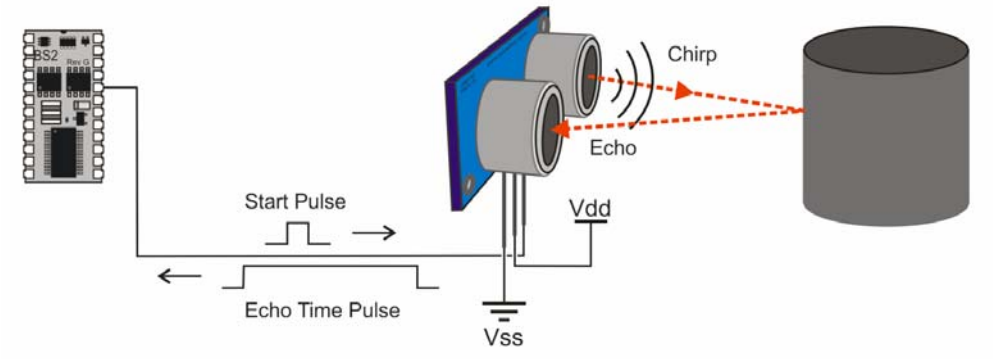
The Ping))) sensor is a device you can use with the BASIC Stamp to measure how far away an object is. With a range of 3 centimeters to 3.3 meters, it's a shoe-in for any number of robotics and automation projects. It's also remarkably accurate, easily detecting an object's distance down to the half centimeter.

Figure 1
The Ping))) Sensor

HOW DOES THE PING))) SENSOR WORK?

Figure 2 shows how the Ping))) sensor sends a brief chirp with its ultrasonic speaker and makes it possible for the BASIC Stamp to measure the time it takes the echo to return to its ultrasonic microphone. The BASIC Stamp starts by sending the Ping))) sensor a pulse to start the measurement. Then, the Ping))) sensor waits long enough for the BASIC Stamp program to start a **`PULSIN`** command. At the same time the Ping))) sensor chirps its 40 kHz tone, it sends a high signal to the BASIC Stamp. When the Ping))) sensor detects the echo with its ultrasonic microphone, it changes that high signal back to low. The BASIC Stamp's **`PULSIN`** command stores how long the high signal from the Ping))) sensor lasted in a variable. The time measurement is how long it took sound to travel to the object and back. With this measurement, you can then use the speed of sound in air to make your program calculate the object's distance in centimeters, inches, feet, etc...

Figure 2 - How the Ping))) Sensor Works



The Ping))) sensor's chirps are not audible because 40 kHz is ultrasonic.

What we consider sound is our inner ear's ability to detect the variations in air pressure caused by vibration. The rate of these variations determines the pitch of the tone. Higher frequency tones result in higher pitch sounds and lower frequency tones result in lower pitch tones.

Most people can hear tones that range from 20 Hz, which is very low pitch, to 20 kHz, which is very high pitch. Subsonic is sound with frequencies below 20 Hz, and ultrasonic is sound with frequencies above 20 kHz. Since the Ping))) sensor's chirps are at 40 kHz, they are definitely ultrasonic, and not audible.

ACTIVITY #1: MEASURING ECHO TIME

In this activity, you will test the Ping))) sensor and verify that it gives you echo time measurements that correspond to an object's distance. You will also modify the example program to convert these times into centimeter measurements.

Parts Required

All you'll need is a Ping))) sensor and three jumper wires to make it work. The Ping))) sensor has protection against programming mistakes (and wiring mistakes) built-in, so there's no need to use a 220 Ω resistor between P15 and the Ping))) sensor's SIG terminal.

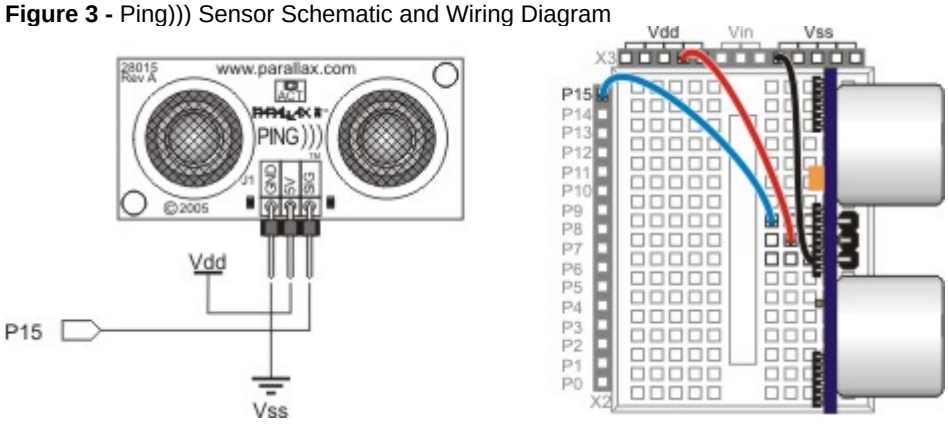
(1) Ping))) Ultrasonic Distance Sensor
(3) Jumper Wires

Ping))) Sensor Circuit

Figure 3 shows a schematic and wiring diagram you can use to test the Ping))) sensor.

√ Build the circuit.

Figure 3 - Ping))) Sensor Schematic and Wiring Diagram



Testing the Ping))) Sensor

As mentioned earlier, the Ping))) sensor needs a start pulse from the BASIC Stamp to start its measurement. A pulse to P15 that lasts 10 µs (**`PULSOUT 15, 5`**) is easily detected by the Ping))) sensor, and it only takes a small amount of time for the BASIC Stamp to send. A **`PULSIN`** command that stores the duration of the Ping))) sensor's echo pulse has to come immediately after the **`PULSOUT`** command. The result the **`PULSIN`** command stores is the round trip time for the Ping))) sensor's chirp to get to the object, reflect and return.

Example Program - PingTest.bs2

You can test this next program by measuring the distances of a few close-up objects. For close up measurements, the Ping))) sensor only needs to be roughly Boe-Bot height above your working surface (8 to 10 cm). However, if you are measuring objects that are more than a half a meter away, make sure to keep your Ping))) sensor about half a meter or more above the floor.

√ Place your Board of Education with the Ping))) sensor circuit on something to keep it at least 8 cm above the table surface.

- √ Place an object (like a water bottle, box, or paper target) 15 cm from the front of the Ping))) sensor.
- √ Enter, save, and run PingTest.bs2.
- √ The Debug Terminal should start reporting a value in the neighborhood of 450. Values of 438 to 466 mean the distance is between 15 and 16 cm.
- √ Move the target to a distance of 30 cm from the Ping))) sensor and verify that the value of the **time** variable doubled.
- √ Point your Ping))) sensor at a variety of near and far objects and observe the time measurements.
- √ Multiply your measurements by 0.03434 to convert to centimeter measurements, and verify that the measurements are correct.

```
' PingTest.bs2

' {$STAMP BS2}
' {$PBASIC 2.5}

time VAR Word

DO

  PULSOUT 15, 5
  PULSIN 15, 1, time
  DEBUG HOME, "time = ", DEC5 time
  PAUSE 100

LOOP
```

Your Turn - Displaying Centimeter Measurements

The next activity will introduce how to derive constants like 0.03434 for converting the echo time measurements to centimeters and other units. But first, let's look at how the PBASIC ****** operator makes it possible to multiply the **time** variable by a value like 0.03434. To convert 0.03434 to a value the ****** operator can use, multiply it by 65536, and use whatever's to the left of the decimal point. Since 0.03434 × 65536 = 2250.5, we'll use 2251 with the ****** operator for the time to centimeter conversion. Here's the conversion statement with the constant we just figured along with a **DEBUG** command to display the centimeter value.

```
  time = time ** 2251
  DEBUG CR, "Distance = ", DEC4 time, " cm"
```

√ Save PingTest.bs2 as PingCentimeters.bs2.

√ Add the two new lines of code to the program's **DO...LOOP** between the **DEBUG** and **PAUSE** commands. When you're done, the **DO...LOOP** should look like this:

```
DO

  PULSOUT 15, 5
  PULSIN 15, 1, time
  DEBUG HOME, "time = ", DEC5 time
  time = time ** 2251
  DEBUG CR, "Distance = ", DEC4 time, " cm"
  PAUSE 100

LOOP
```

√ Run your modified program and verify that the program correctly displays both the echo time and centimeter measurements.