Chapter 2: The Ping))) Ultrasonic Distance Sensor

2

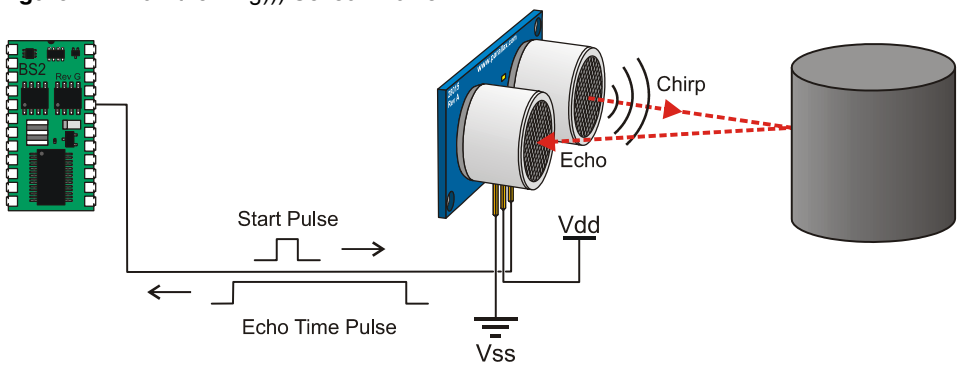
The Ping))) sensor interfaced with a BASIC Stamp can measure how far away objects are. With a range of 3 centimeters to 3.3 meters, it's a shoo-in for any number of robotics and automation projects. It's also remarkably accurate, easily detecting an object's distance down to the centimeter.

Figure 2-1
The Ping))) ™ Ultrasonic Distance Sensor

HOW DOES THE PING))) SENSOR WORK?

Figure 2-2 shows how the Ping))) sensor sends a brief chirp with its ultrasonic speaker and measures the echo's return time 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. Then, 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.

Figure 2-2: How the Ping))) Sensor Works



The BASIC Stamp **PULSIN** command uses a variable to store how long the high signal from the Ping))) sensor lasted. This time measurement is how long it took sound to travel to the object and back. Using this measurement and the speed of sound in air, you can make your program calculate the object's distance in centimeters, inches, feet, etc.

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 to people.

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.

Parts Required

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

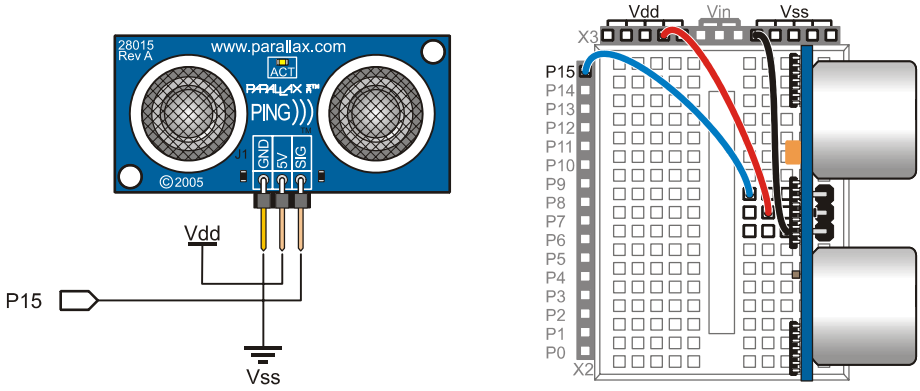
All you'll need is a Ping))) sensor and three jumper wires to make it work. The Ping))) sensor has protection against programming mistakes (and wiring mistakes) built-in, so there's no need to use a 220 Ω resistor between P15 and the Ping))) sensor's SIG terminal.

Ping))) Sensor Circuit

Figure 2-3 shows a schematic and wiring diagram for testing the Ping))) sensor.

√ Build the circuit.

Figure 2-3: 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 (**`PULSIN 15, 1, time`**) has to come immediately after the **`PULSOUT`** command. In this example, the result the **`PULSIN`** command stores in the variable **`time`** 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 3 to 4 inches (roughly 8 to 10 cm) above your working surface. However, if you are measuring objects that are more than a half a meter away, you may need to elevate your Ping))) sensor to prevent echoes from the floor registering as detected objects.

- √ 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 400 to 500 range.

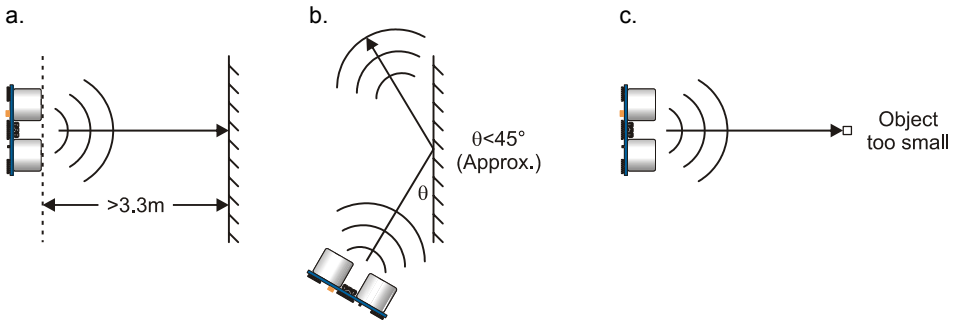
- √ Move the target to a distance of 30 cm from the Ping))) sensor and verify that the value of the time variable roughly doubled.
- √ Point your Ping))) sensor at a variety of near and far objects, and observe the time measurements.

```
' Smart Sensors and Applications - PingTest.bs2
' Tests the Ping))) ultrasonic distance sensor

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

time VAR Word

DO
  PULSOUT 15, 5
  PULSIN 15, 1, time

  DEBUG HOME, "time = ", DEC5 time

  PAUSE 100
LOOP
```

Your Turn - Testing Range, Angle and Object Size

In terms of accuracy and overall usefulness, ultrasonic distance detection is really great, especially compared to other low-cost distance detection systems. That doesn't mean that the Ping))) sensor is capable of measuring "everything". Figure 2-4 shows a few situations that the Ping))) is not designed to measure: (a) distances over 3 meters, (b) shallow angles, and (c) objects that are too small.

Figure 2-4: The Ping))) Sensor is Not Designed for these Situations:



In addition, as Ken Gracey of Parallax Inc. discovered during a classroom demonstration at his son's school, some objects with soft, irregular surfaces (such as stuffed animals) will absorb rather than reflect sound and therefore can be difficult for the Ping))) sensor to detect. Objects with smooth surfaces that readily reflect sound are easier for the sensor to detect.

- √ Try pointing the Ping))) sensor at various objects that are different distances away. What's the largest value the Ping))) sensor returns? How close do you have to get to the object before the time measurements start to decrease?
- √ Try standing one meter away from the wall, and point the Ping))) sensor at it, and record the measurement. Next, try pointing the Ping))) sensor at the wall at different angles, as shown in Figure 2-5. Do the values change? At what angle does the Ping))) sensor cease to detect the wall?

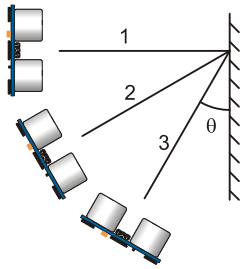


Figure 2-5
Determining the Minimum Angle of Detection

- √ Try hanging various objects from the ceiling at about 1.5 meters from the Ping))) sensor. How small can the object be? Does shape or angle matter? Does the size requirement change at 0.5 meters?
- √ Try detecting objects of similar size but made from different materials, such as a cardboard shoebox and a fuzzy slipper, to see if you have a smaller effective range with sound-absorbing objects. Can you find any objects invisible to the Ping))) sensor? How about a wad of cotton balls, or of tulle netting?

ACTIVITY #2: CENTIMETER MEASUREMENTS

This activity demonstrates how to use the speed of sound and the PBASIC Multiply High operator (`**`) to calculate the distance of an object based on the echo time measurement from the Ping))) sensor.

Calculating Centimeter Distance with PBASIC

The equation for the distance sound travels is $S = C_{air}\ t$, where S is distance, C_{air} is the speed of sound in air, and t is time. Since the Ping))) sensor's time measurement is the time it takes sound to get to the object and bounce back, the actual distance, S_{object}, is half of the total distance the sounds travels.

$$S = C_{air} t$$

$$S_{object} = \frac{S}{2} = \frac{C_{air} t}{2}$$

The speed of sound in air is most commonly documented in terms of meters per second (m/s). However, centimeter (cm) measurements will be more convenient to calculate with the BASIC Stamp. Since there are 100 centimeters in a meter, let's use $S_{object\text{-}cm}$ which is simply 100 times S_{object}. The **`PULSIN Duration`** measurement units for the BASIC Stamp 2 are 2/1,000,000 of a second (2 μs). So, instead of t, which has to be a measurement of seconds, we'll use $t_{PULSIN\text{-}BS2}$. When multiplied by 2/1,000,000 $t_{PULSIN\text{-}BS2}$ gives us the number of seconds. There is a pair of 2s in the numerator and denominator that cancel, and 100 in the numerator can cancel with two of the zeros in the denominator's 1,000,000. The result of these substitutions and cancellations is $S_{object\text{-}cm} = (C_{air}\ t_{PULSIN\text{-}BS2})/10{,}000$.

$$S_{object\text{-}cm} = \frac{100\, C_{air} t}{2}$$

$$S_{object\text{-}cm} = \frac{100\, C_{air} t_{PULSIN\text{-}BS2}}{2} \times \frac{2}{1{,}000{,}000}$$

$$S_{object\text{-}cm} = \frac{C_{air} t_{PULSIN\text{-}BS2}}{10{,}000}$$

The speed of sound in air at room temperature 72 °F (22.2 °C) is 344.8 m/s. Dividing 10,000 into this leaves us with $S_{object\text{-}cm} = 0.03448\ t_{PULSIN\text{-}BS2}$.

$$S_{object\text{-}cm} = \frac{344.8\ t_{PULSIN\text{-}BS2}}{10{,}000}$$

$$= 0.03448\ t_{PULSIN\text{-}BS2}$$

The BASIC Stamp can use the ****** operator to multiply a variable that stores the **PULSIN** command's ***Duration*** measurement by a fractional value that's less than 1. For example, if the **PULSIN** command stores the echo measurement in the **time** variable, this command will store the centimeter distance result in the **cmDistance** variable:

```
cmDistance = CmConstant ** time
```

With the ****** operator, **CmConstant** will have to be 2260, which is the ****** equivalent of 0.03448. Instead of a decimal denominator, like 10,000 (in the case of 0.03448), the ****** operator needs a value that would go in the numerator of a fraction with a denominator of 65536. To get that numerator, multiply your fractional value by 65536.

$$CmConstant = 0.03448 \times 65536 = 2260$$

Now, we've got the value we need to modify PingTest.bs2 so that it will measure centimeter distance. We'll also add a variable to store distance (**cmDistance**) along with the constant to store the value 2260 (**CmConstant**).

```
CmConstant  CON   2260

cmDistance  VAR   Word
```

Then, the ****** calculation can be added to the PingText.bs2's **DO**...**LOOP** to calculate the centimeter measurement. The **DEBUG** command in the program can then be modified to display the measurement.

```
cmDistance = CmConstant ** time

DEBUG HOME, DEC3 cmDistance, " cm"
```

Example Program: PingMeasureCm.bs2

- √ Enter, save and run PingMeasureCm.bs2.
- √ Move the target object until the measurement displays 20 cm.
- √ Align your ruler with that measurement. The 0 cm mark should align somewhere with the Ping))) sensor, typically somewhere between the printed circuit board and the front-most part of the speaker/microphone.
- √ Now, experiment with other distance measurements.

```
' Smart Sensors and Applications - PingMeasureCm.bs2
' Measure distance with Ping))) sensor and display in centimeters.

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

' Conversion constants for room temperature measurements.
CmConstant  CON   2260

cmDistance  VAR   Word
time        VAR   Word

DO

  PULSOUT 15, 5
  PULSIN 15, 1, time

  cmDistance = CmConstant ** time

  DEBUG HOME, DEC3 cmDistance, " cm"

  PAUSE 100

LOOP
```

Your Turn - Verifying the Calculations

Let's verify that that the program is correctly calculating the distance.

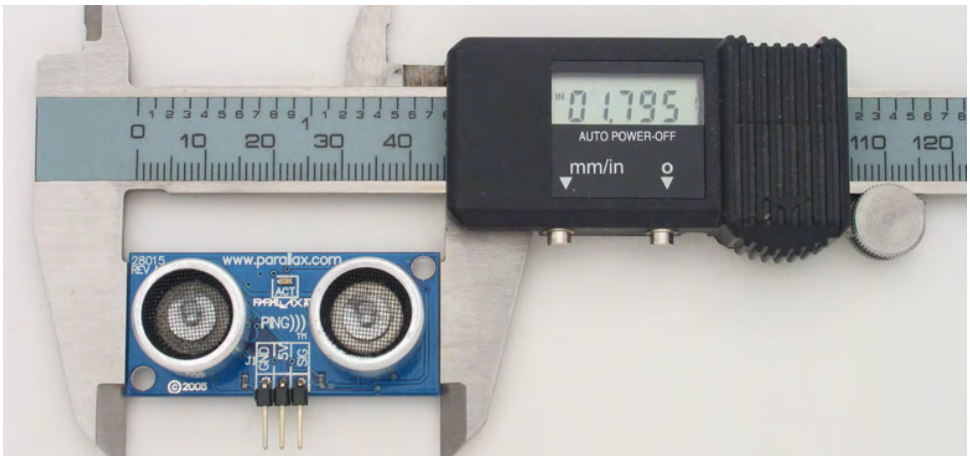
- √ Modify PingMeasureCm.bs2 so that it displays the values of both the time and the distance variables.
- √ Use a calculator to verify that you get the same result from the distance equation that you do from the program.

$$S_{object\text{-}cm} = 0.03448 \times t_{PULSIN\text{-}BS2}$$

ACTIVITY #3: INCH MEASUREMENTS

Most electronic distance measuring devices offer results in either metric or English units. For example, the calipers shown in Figure 2-6 has a button you can press to choose between mm and inches. Other measuring devices offer yards or meters, or inches or centimeters, etc. So that your program can display both centimeters and inches, this activity introduces uses the multiply-high (`**`) operator a second time to convert from centimeters to inches.

Figure 2-6: Calipers with a mm/in Toggle Button



An Inches ** Constant

The **`CmConstant`** used in **`cmDistance = time ** CmConstant`** is a measure of the speed of sound in centimeters per **`PULSOUT`** time unit. There are 2.54 centimeters in every inch. So, the conversion formula from centimeter to inch distances can be written like this:

$$S_{in} = S_{cm} \div 2.54$$

The easiest way to convert to inches is to simply divide the value of **`CmConstant`** by 2.54, and use the result declared in another constant, like **`InConstant`**. Remember that constants for the `**` operator should be integers, so round the result to the nearest integer.

$$InConstant = 2260 \div 2.54 = 889.76 \approx 890$$

Example Program: PingMeasureCmAndIn.bs2

- √ Enter, save, and run PingMeasureCmAndIn.bs2.
- √ Experiment with the distance measurements and verify that they are correct in both systems.

```
' 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
```

Your Turn

- √ There are 12 inches in 1 foot. Modify the program so that it displays feet and inches. Hint: After calculating **`inDistance`**, use **`/ 12`** to figure out the number of feet, and **`// 12`** to find the remainder in inches.
- √ There are 10 centimeters in a decimeter. Repeat for decimeters and centimeters.

ACTIVITY #4: MOBILE MEASUREMENTS

This activity demonstrates displaying the Ping))) sensor's centimeter and inch measurements on the Parallax Serial LCD. Provided you're using a battery, you can disconnect from your computer and take the setup to remote locations of your choosing.

Connecting the Ping))) Sensor with an Extension Cable

In order to make room for the Parallax Serial LCD on the Board of Education, we'll connect the Ping))) sensor to the board with an extension cable. You can then hold it and point it various places, or use hardware to mount it next to your Board of Education.

Parts Required

(1) Ping))) Ultrasonic Sensor
(1) Parallax Serial LCD (2×16)
(1) 14-inch LCD Extension Cable
(3) Jumper Wires

If you are working from a BASIC Stamp HomeWork Board or a serial Board of Education Rev A or B, you will also need:

(1) 3-pin header
(3) Additional jumper wires

Ping))) Sensor and LCD Cable Connections

The schematics shown in Figure 2-7 below are identical to the ones that have been used for the Ping))) sensor and the Parallax Serial LCD up to this point. We will now change the way these electrical connections are made by adding a cable, so that both devices can be conveniently connected to your board at the same time. Though the schematics are the same, the actual cable connections will vary depending on which BASIC Stamp educational board you are using.

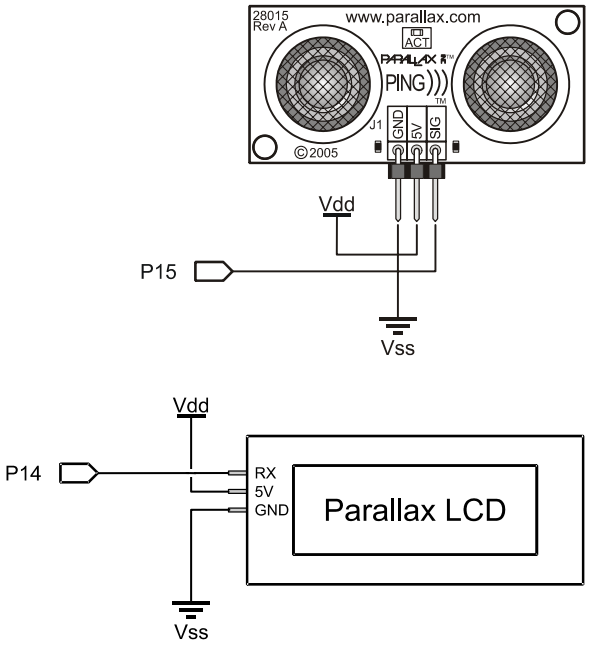


Figure 2-7
Ping))) Sensor and Parallax Serial LCD Schematic

Board of Education Rev C and USB Board of Education Cable Connections

These instructions are for the boards that have servo ports with a Vdd/Vss jumper in between, such as the Board of Education Rev C and USB Board of Education. For all other boards, skip to **All other BASIC Stamp Educational Boards** on page 54.

- √ Disconnect power to your board.
- √ Set the jumper between the X4 and X5 servo to Vdd (+5 V) as shown in Figure 2-8. The jumper should cover the two pins closest to Vdd, and the third pin next to Vin should be visible.

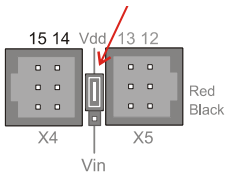


Figure 2-8
The Servo Port Jumper Set to Vdd (+5 V)

Vdd vs. Vin jumper settings determine which power supply is connected to the X4 and X5 ports. When the jumper is set to Vdd, these ports receive regulated 5 V from the Board of Education's voltage regulator. If the jumper is set to Vin, the port receives power directly from the battery or power supply.

- √ Connect the Parallax Serial LCD as shown. It's the same as the previous chapter.
- √ Plug one end of the extension cable into Port 15 of the X4 header, making sure that the "Red" and "Black" labels along the right side of the X5 port line up with the cable's red and black wires.
- √ Verify that your cable is plugged in correctly by checking to make sure the white wire is closest to the 15 label and the black wire is closest to the X4 label.

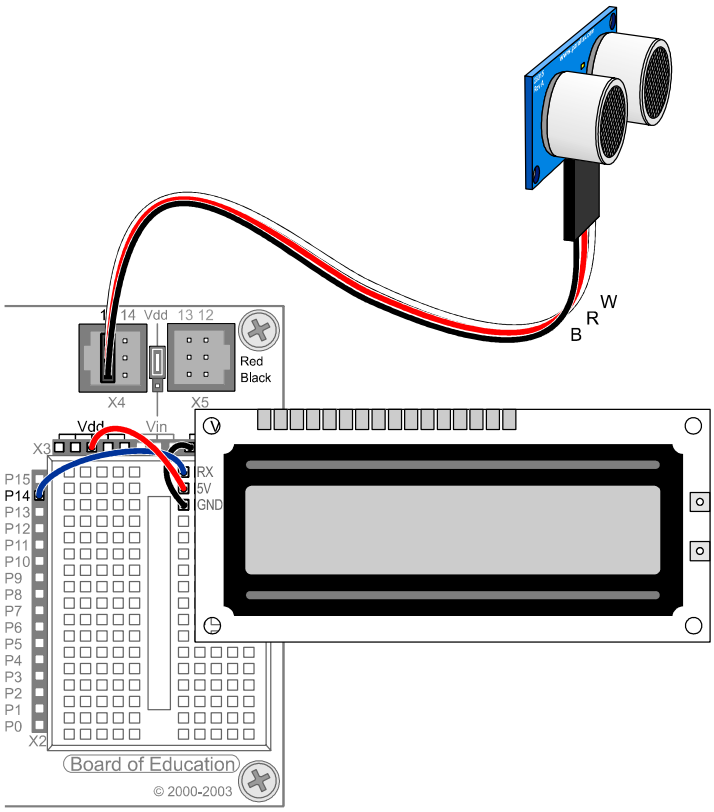


Figure 2-9
Servo Port and Power Jumper Connection for Ping))) Sensor

√ Connect the other end of the cable so that the black wire is connected to the Ping))) module's GND pin, the red wire is connected to the 5 V pin, and the white wire is connected to the RX pin.
√ Double-check all your connections, including your jumper setting, and make sure they are correct.

WARNING! Do not connect power to your board until you are positive the connections are correct. If you make a mistake with the LCD connections, the Parallax Serial LCD could be permanently damaged.

√ Plug the power back into the board.
√ Set the 3-position switch on the Board of Education to 2.
√ If you have a Board of Education Rev C, Skip to **LCD Distance Display** on page 57.

You can also connect the Parallax Serial LCD to Port 14 with a cable. The instructions are about the same as connecting the Ping))). Start by disconnecting power to your board. The jumper for Vdd and Vin between the servo ports has to be set to Vdd. The cable has to be plugged into the X4 header so that the black wire is closest to the X4 label and the white wire is closest to the 14 label. When connecting the other end of the cable to the Parallax Serial LCD, make sure the black wire connects to GND, the red wire to 5V, and the white wire to RX.

All other BASIC Stamp Educational Boards

This section is for connecting the Ping))) sensor and Parallax Serial LCD to one of the following BASIC Stamp educational boards:

- BASIC Stamp HomeWork Board
- Board of Education Rev A (Serial version)
- Board of Education Rev B (Serial version)

√ Disconnect power from your board.
√ Build the breadboard connections as shown in Figure 2-10.

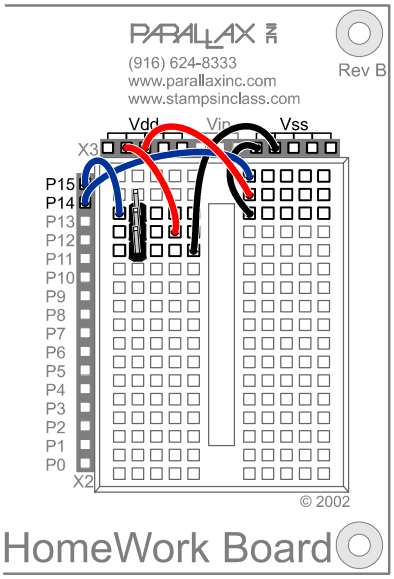


Figure 2-10
Breadboard Wiring for Ping)))
Sensor Cable Connection

√ Plug the Parallax Serial LCD into the breadboard as shown in Figure 2-11 on page 56.

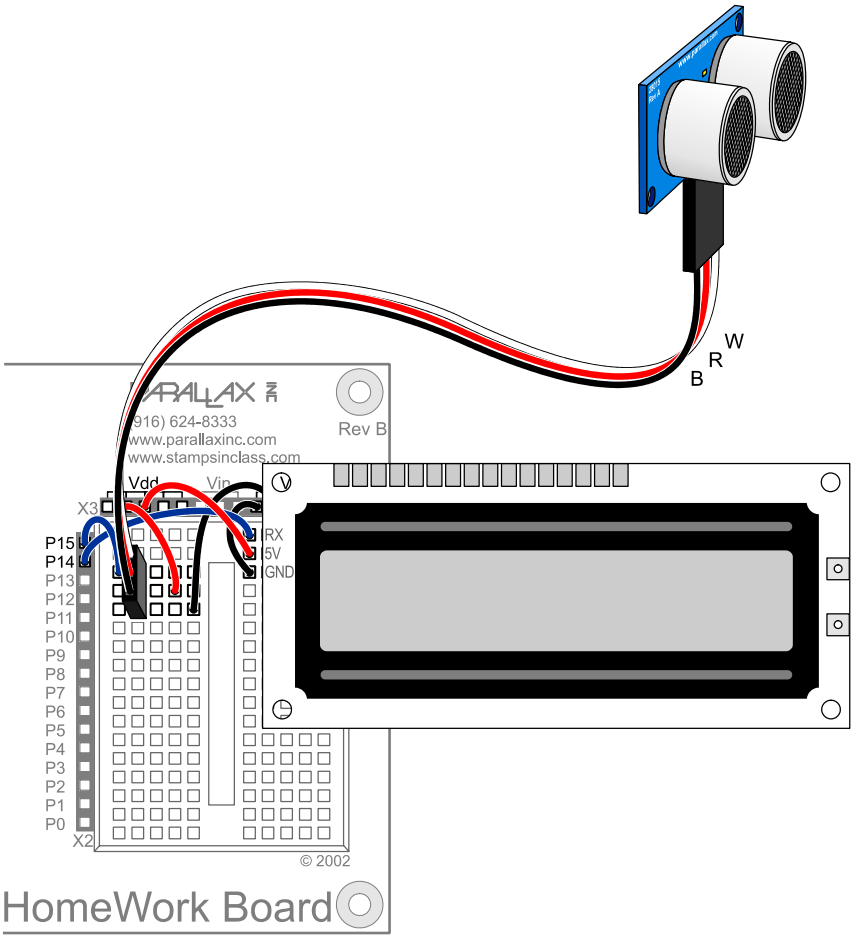
√ Plug one end of the extension cable into the 3-pin header, making sure the white, red, and black wires are oriented as shown. The black wire should be connected to Vss, the Red wire to Vdd, and the white wire to P15.

√ Connect the other end of the cable so that the black wire is connected to the Ping)))'s GND pin, the red wire is connected to the 5 V pin, and the white wire is connected to the RX pin. Double-check all your connections, including your jumper setting, and make sure they are correct.

WARNING! Do not connect power to your board until you are positive the connections are correct. If you make a mistake with the LCD connections, the Parallax Serial LCD could be permanently damaged.

√ Reconnect power to your board.

Figure 2-11: Breadboard Connections for Ping))) Sensor and Parallax Serial LCD



LCD Distance Display

It doesn't take much in the way of changes to modify PingMeasureCmAndIn.bs2 to make it display its measurements on the LCD. First, an Initialization section has to be added so that the program waits for the power supply to stabilize and then turns on and clears the LCD.

```
PAUSE 200
SEROUT 14, 84, [22, 12]
PAUSE 5
```

Next, the **DEBUG** commands need to be changed to **SEROUT** commands. Here are the **DEBUG** commands from PingMeasureCmAndIn.bs2.

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

The Debug Terminal control characters (**HOME** and **CR**) need to be changed to control codes that will place the cursor in the LCD.

```
SEROUT 14, 84, [128, DEC3 cmDistance, " cm"]
SEROUT 14, 84, [148, DEC3 inDistance, " in"]
```

Example Program: PingLcdCmAndIn.bs2

This program is a modified version of PingMeasureCmAndIn.bs2 from the previous activity. Instead of displaying its measurements in the Debug Terminal, it displays them on the Parallax Serial LCD.

- √ Connect a battery to your board.
- √ Enter, save and run PingLcdCmAndIn.bs2.
- √ Disconnect the serial cable, and take your board with you to wherever you want to test the Ping))) sensor's measurements.

```
' Smart Sensors and Applications - PingLcdCmAndIn.bs2
' Measure Distance with the Ping))) sensor and display on LCD

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

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

cmDistance  VAR   Word
```

```
inDistance  VAR   Word
time        VAR   Word

PAUSE 200
SEROUT 14, 84, [22, 12]
PAUSE 5
DEBUG CLS, "Program running..."

DO

  PULSOUT 15, 5
  PULSIN 15, 1, time

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

  SEROUT 14, 84, [128, DEC3 cmDistance, " cm"]
  SEROUT 14, 84, [148, DEC3 inDistance, " in"]

  PAUSE 100

LOOP
```

Your Turn - Customizing the Display

- √ The measurements are currently left-justified. Try centering them.
- √ Try right justifying the measurements and displaying "Distance: " before the cm measurement on the LCD's top line.
- √ Modify the program so that it displays both of the distance measurements on the top line. Then, display the actual echo time on the bottom line. You can display it in millionths of a second (μs) by multiplying the time variable by 2 before displaying it. Make sure your program waits until after it has done its distance conversions before multiplying time by 2.

ACTIVITY #5: TEMPERATURE'S EFFECT ON THE SPEED OF SOUND

This activity investigates changes in the speed of sound caused by changes in air temperature. These changes in the speed of sound can result in visible changes to your distance measurements.

Speed of Sound Vs Temperature and Percent Error Measurements

The speed of sound changes with air temperature, humidity, and even air quality. Neither humidity nor air quality make enough of a difference to figure into Ping))) sensor

distance calculations. Air temperature, on the other hand, can cause measurable distance errors. The speed of sound increases by 0.6 meters per second (m/s) for every degree-Celsius (°C) increase in temperature. Since the speed of sound is about 331.5 m/s at 0 °C, we can use this equation to calculate the speed of sound at a given temperature.

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

Converting from °F to °C and Visa Versa

To convert a degree-Fahrenheit to Celsius, subtract 32 from T_F (the Fahrenheit measurement), then divide by 1.8. The result will be T_C, the Celsius equivalent. To convert from Celsius to Fahrenheit, multiply T_C by 1.8, then add 32. The result will be T_F.

$$T_C = (T_F - 32) \div 1.8 \qquad T_F = 1.8 \times T_C + 32$$

Below are examples for the speed of sound at two fairly comfortable, but slightly different indoor temperatures.

Example 1: Calculate the speed of sound at 22.2 °C, which is approximately 72 degrees Fahrenheit (°F).

$$C_{air}\ (22.2°C) = 331.5 + (0.6 \times 22.2\ m/s) = 344.8\ m/s$$

Example 2: Calculate the speed of sound at 25 °C, which is 77 degrees Fahrenheit (°F).

$$C_{air}\ (25°C) = 331.5 + (0.6 \times 25)\ m/s = 346.5\ m/s$$

How much of a difference does this make to your distance measurements? We can calculate the percent error this will propagate with the percent error equation.

$$\%\ error = \frac{actual - predicted}{predicted} \times 100\%$$

If the predicted temperature in the room is 72 °F (22.2 °C), and the actual temperature is 77 °F (25 °C), the error is 0.49 percent. Half a percent error would cause you to have to

move the object half a centimeter beyond 100 cm before it would transition from 99 to 100 cm.

$$\%\,\text{error} = \frac{346.5 - 344.8}{344.8} \times 100\%$$

$$= 0.49\%$$

Your Turn - Room Temperature Vs. Freezing

- √ Calculate the percent measurement error that would result from assuming that the ambient temperature is freezing (32 °F, 0 °C), but it's actually room temperature (72 °F, 22.2 °C).
- √ How far off would the measurement be if the object is 1 m away?
- √ Use the procedure introduced in Activity #2 to calculate the speed of sound and **`CmConstant`** for measurements at 0 °C.
- √ Save PingMeasureCm.bs2 as PingMeasureCmYourTurn.bs2
- √ Run the program before modifying it and test the distance measurement of an object at 1 m.
- √ Modify the **`cmConstant CON`** directive with the value for 0 °C.
- √ Re-test the program with an object at 1 m. How close is your predicted error to the actual error?

SUMMARY

The BASIC Stamp requests a measurement from the Ping))) sensor by sending it a brief pulse, which causes it to emit a 40 kHz chirp. Then, the Ping))) listens for an echo of that chirp. It reports the echo by sending a pulse back to the BASIC Stamp that is equal to the time it took for the Ping))) sensor to receive the echo.

To calculate the distance based on the echo time measurement, the speed of sound must be converted into units that are convenient for the BASIC Stamp. This involves converting meters per second to centimeters per **PULSIN** measurement units. The resulting value also has to be converted to a value that can be used with the multiply high (**) operator by multiplying it by 65536.

The speed of sound in air is $c_{air} = 331.5 + (0.6 \times T_C)$ m/s. While the speed of sound changes with temperature, the resulting measurement errors are small, especially at room temperature.

Questions

1. What is the Ping))) sensor's range?
2. What does ultrasonic mean?
3. What signal does the Ping))) sensor send the BASIC Stamp and how does it correspond to a distance measurement?
4. What three sensor-object orientation scenarios could cause the Ping))) sensor to return an incorrect distance measurement?
5. What time increments does the **PULSIN** command return when using a BS2?
6. What's the speed of sound in air at room temperature?
7. How does **CmConstant** relate to the speed of sound in air?
8. What do you have to do to the jumper between the X4 and X5 servo headers on the Board of Education to provide the correct supply voltage to devices like the Ping))) sensor and the Parallax Serial LCD? What might happen if this jumper is not correctly set?
9. What commands have to be modified if you want to make the Parallax LCD display what the Debug Terminal was displaying?
10. What role does air temperature play in the speed of sound in air?

Exercises

1. Calculate how many meters away an object is if the echo time is 15 ms, and the temperature is 22.5 °C.
2. Calculate the °C equivalent of 100 °F.
3. Calculate the foot equivalent of 30.48 cm.
4. Calculate the percent error if **`CmConstant`** is for 37.8 °C but the temperature is 0 °C. Predict what the measured distance would be if the object were placed at 0.5 m.

Projects

1. Add an LED circuit to your board and program the BASIC Stamp to make the LED flash when there is no object in range.
2. Use a piezospeaker to make an alarm that signals when people pass through a doorway. The Ping))) sensor should be mounted next to the doorway, pointing across the path people walk when they enter or leave.

Solutions

Q1. 3 centimeters to 3.3 meters.
Q2. Sound with frequencies above 20 kHz.
Q3. A high pulse, whose duration corresponds to the time it took for the chirp sound to travel to the object and back.
Q4. a) Distance over 3 meters, b) Shallow angles, c) Objects that are too small.
Q5. 2μs increments.
Q6. 344.8 m/s.
Q7. **`CmConstant`** is the `**` equivalent of the speed of sound in air divided by 10000, or 0.03448.
Q8. The jumper should be set to the Vdd position, otherwise the LCD could be damaged.
Q9. All **`DEBUG`** commands have to be modified, and control characters have to be modified to use the LCD's control codes.
Q10. A very important role, with the speed increasing 0.6 m/s for every degree C increase in air temperature.
E1. The object is 2.59 m away.
E2. 100 °F = 37.7 °C
E3. 30.48 cm = 1.0 ft.
E4. % error = +/- 6.84%; measured distance = 0.466 m.
P1. This example solution places an active-high LED on P13.

```
' Smart Sensors and Applications - Ch2_Project1.bs2
' Indicate out-of-range with flashing LED.  Adjust MaxDistance to suit.
' {$STAMP BS2}
' {$PBASIC 2.5}

LED             PIN     13              ' Red LED active high
LCD             PIN     14              ' Parallax Serial LCD
Ping            PIN     15              ' Parallax Ping))) sensor

CmConstant      CON     2260            ' Calc roundtrip time of sound
InConstant      CON     890
MaxDistance     CON     361             ' Maximum can measure (empirical)
cmDistance      VAR     Word            ' Distance in centimeters
time            VAR     Word            ' Round trip echo time

PAUSE 200                               ' Initialize LCD
SEROUT LCD, 84, [22, 12]
PAUSE 5

DO
  LOW LED                               ' LED off before each measurement
  PULSOUT 15, 5                         ' Start Ping)))
```

```
  PULSIN 15, 1, time                 ' Read echo time
  cmDistance = cmConstant ** time    ' Calculate distance from time
  SEROUT LCD, 84, [128, DEC3 cmDistance, " cm"]  ' Print distance on
                                                 ' LCD scrn
  IF cmDistance >= MaxDistance THEN HIGH LED     ' Toggle LED if out
                                                 ' of range
  PAUSE 100
LOOP
```

P2. Example solution:

```
' Smart Sensors and Applications - Ch2_Project2.bs2
' Make a sound when someone passes through the doorway.

' {$STAMP BS2}
' {$PBASIC 2.5}
' -----[ I/O Definitions ]-------------------------------------------
Ping           PIN     15            ' Parallax Ping))) sensor
Speaker        PIN     9             ' Optional speaker

' -----[ Constants ]-------------------------------------------------
InConstant     CON     890
Doorjamb       CON     35            ' Doorway width is 35 inches

' -----[ Variables ]-------------------------------------------------
inDistance     VAR     Word
time           VAR     Word          ' Round trip echo time
counter        VAR     Nib

' -----[ Main Routine ]----------------------------------------------
DO
  GOSUB Read_Ping
  GOSUB Calc_Distance
  IF (inDistance < Doorjamb) THEN
    GOSUB Sound_Alarm
  ENDIF
LOOP
' -----[ Subroutines ] ----------------------------------------------
Read_Ping:
  PULSOUT 15, 5                      ' Start Ping)))
  PULSIN 15, 1, time                 ' Read echo time
  RETURN

Sound_Alarm:
  FREQOUT Speaker, 300, 3300         ' Bing
  PAUSE 50
  FREQOUT Speaker, 450, 2200         ' Bong
  RETURN

Calc_Distance:
  inDistance = inConstant ** time    ' These are the whole measurements
  RETURN
```