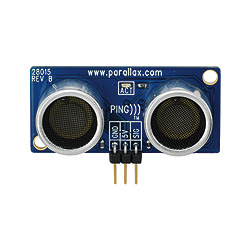
**Sensor User Manual**

Last updated: 27 September 2012

Prepared by Marco De Mutiis [me@marcodemutiis.com]

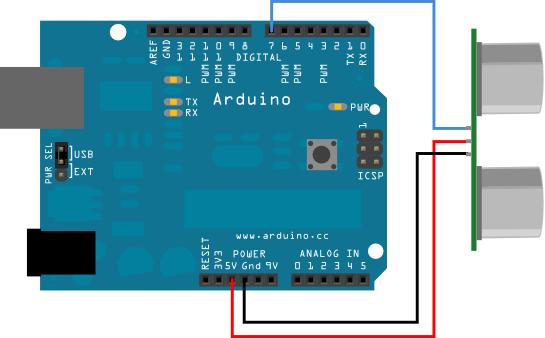
Sensor: Parallax PING))) Ultrasonic Distance Sensor

URL: http://www.parallax.com/tabid/768/ProductID/92/Default.aspx  
Cost per unit: $29.99 USD



General Information  
  
**About Ultrasound**  
  
Ultrasound is a cyclic sound pressure wave with a frequency greater than the upper limit of human hearing. Ultrasound is thus not separated from "normal" (audible) sound based on differences in physical properties, only the fact that humans cannot hear it. Although this limit varies from person to person, it is approximately 20 kilohertz (20,000 hertz) in healthy, young adults.   
  
Many animals—such as dogs, cats, dolphins, bats, and mice—have an upper frequency limit that is higher than that of the human ear and thus can hear ultrasound. This is why a dog whistle can be heard by a dog (excerpted from wiki article on ultrasound, accessed 27 September 2012).  
  
**About Ultrasonic Sensors**  
  
It detects the distance of the closest object in front of the sensor (from 2 cm up to 3m). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the same pin using the pulseIn() function. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance (excerpted from arduino playground, accessed 27 September 2012).  
  
**Measuring Distance – the Math**  
  
According to Parallax’s datasheet ultrasonic waves take 73.746 microseconds to travel 1 inch of distance (or, 1130 feet per second); or, 340 m/s, or 29 microseconds per 1 cm. The pulseIn() function gives the time taken for waves to travel from sensor to obstacle and back to sensor- so we divide this number by 2 to get distance travelled; see example code.  
  
**Features & Specifications**  
  
- Provides precise, non-contact distance measurements within a 2 cm to 3 m range   
- Simple pulse in/pulse out communication   
- Burst indicator LED shows measurement in progress   
- 20 mA power consumption   
- Narrow acceptance angle   
- 3-pin header makes it easy to connect using a servo extension cable  
- Power requirements: +5 VDC   
- Communication: Positive TTL pulse   
- Dimensions: 0.81 x 1.8 x 0.6 in (22 x 46 x 16 mm)   
- Operating temp range: +32 to +158 °F (0 to +70 °C)

Getting it to work!  
  
**Connection**



5V of the PING))) connected to 5V pin on Arduino  
GNC of the PING))) connected to GND on Arduino  
SIG of the PING))) connected to Any digital input pin (example uses 7)  
  
**Example Code**



Code: in plain English

long hellokitty, inches, cm;  
  
*Establish variables for duration of the ping. Gives result in inches and centimeters. We use the "long" variable type because the values returned by sensor are likely to be large.*

pinMode(pingPin, OUTPUT);  
digitalWrite(pingPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(pingPin, LOW);

*Tell Arduino to treat pin 7 as an OUTPUT. We send out a HIGH pulse of 10 microseconds to pin 7. We then switch off the pulse.*

pinMode(pingPin, INPUT);  
hellokitty = pulseIn(pingPin, HIGH);

*The "pulseIn" function: this function reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Function returns the length of the pulse in microseconds. Just for fun I am calling the PING's duration "hellokitty."*

inches = microsecondsToInches(hellokitty);  
cm = microsecondsToCentimeters(hellokitty);

*Convert time into distance, we specify these operations at the end of the code.*

long microsecondsToInches(long microseconds)  
{  
return microseconds / 74 / 2;  
}

*73.746 microseconds per inch (1130 feet per second). This gives the distance travelled by the ping, outbound and return, so we divide it by 2 to get the distance of the obstacle.*

{  
return microseconds / 29 / 2;  
}

*The speed of sound is 340 m/s or 29 microseconds per centimeter. The ping travels outward and back.*