Experiment 3: Sonar Navigation, CAD and 3D Printing

V3 Robot scans area for obstacles to avoid hitting them and navigates using sonar sensor mounted on a 3D printed sensor bracket that goes on a micro servo on the upper chassis of your robot. You will design and print your sensor bracket (only if you have a 3D printer available otherwise use the sonar sensor bracket provided with your kit).

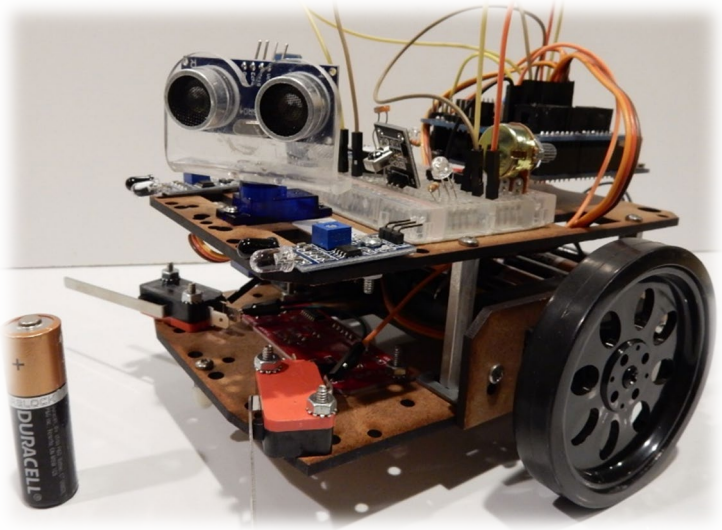
Purpose:

Navigate with sonar sensor that scans the area for obstacles on a 180 degree range and when the robot encounters an obstacle it backs away and picks another direction to go. This will involve a third servo where the sonar will be mounted and also you print your own sonar sensor bracket using the 3D printer.
Those of you taking the class in person must show that the robot can scan in front of it then take the path that is open to go thru it. For example you will have multiple items in front of the robot, the robot will stop and scan the area then head towards the opening area of the obstacle course. The other requirement is that you design and 3D print your own bracket. You may use the open SCAD program or your own CAD program that can export as an STL file format.

What you will learn:

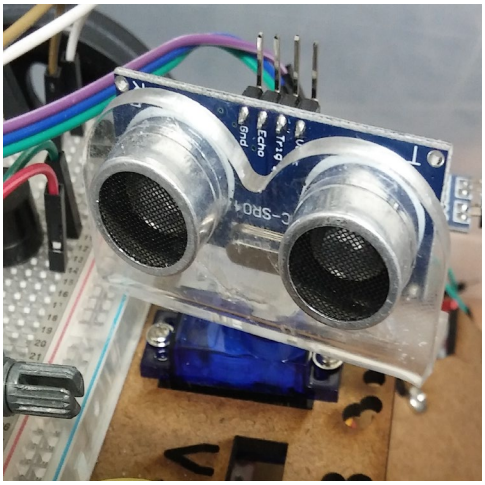
How to program a different kind of sensor for autonomous navigation. Sonar is based on sound waves so you will see how this sensor differs from the feelers (plain switch) and infrared sensors. How to make the Arduino run multiple tasks using timers – scanning the surrounding and detecting while roaming using Timers.
Good tutorial on Timers and Interrupts can be read here: http://letsmakerobots.com/content/arduino-101-timers-and-interrupts , but Timers essentially let you run multiple tasks at the "same" time by running them at particular time sequence. You might think the Arduino is doing multiple things at once, but is really separating each functions by slicing the time it dedicates to it. It is so fast that a human eye can't even notice it.

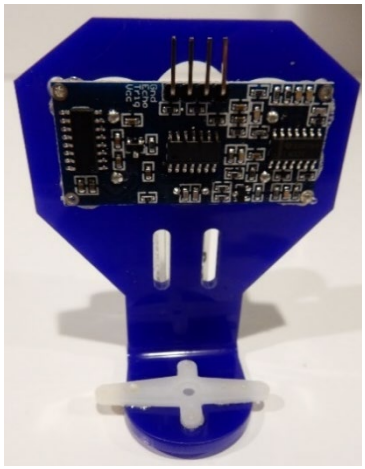
You will also learn how to draw in 3D using an open source line programming based 3D computer aided drafting program called SCAD, how to export the drawing you created as an STL file (readable format by 3D printer software) and use if you have a 3D printer to print your very own sensor bracket.

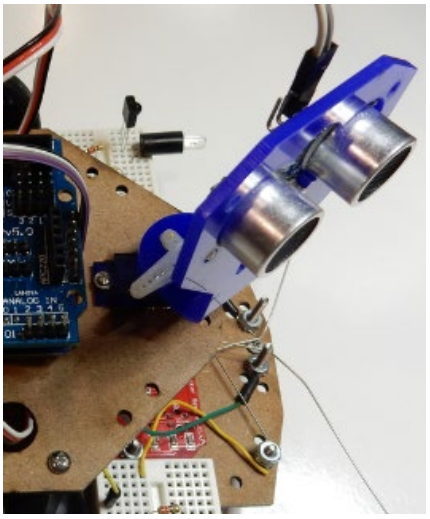


CAUTION: If you connect the sensor wrong you will burn it! Pay attention to the polarity.

Sample Sonar Sensor and Acrylic Brackets







How to design a part, export as STL and 3D print the part

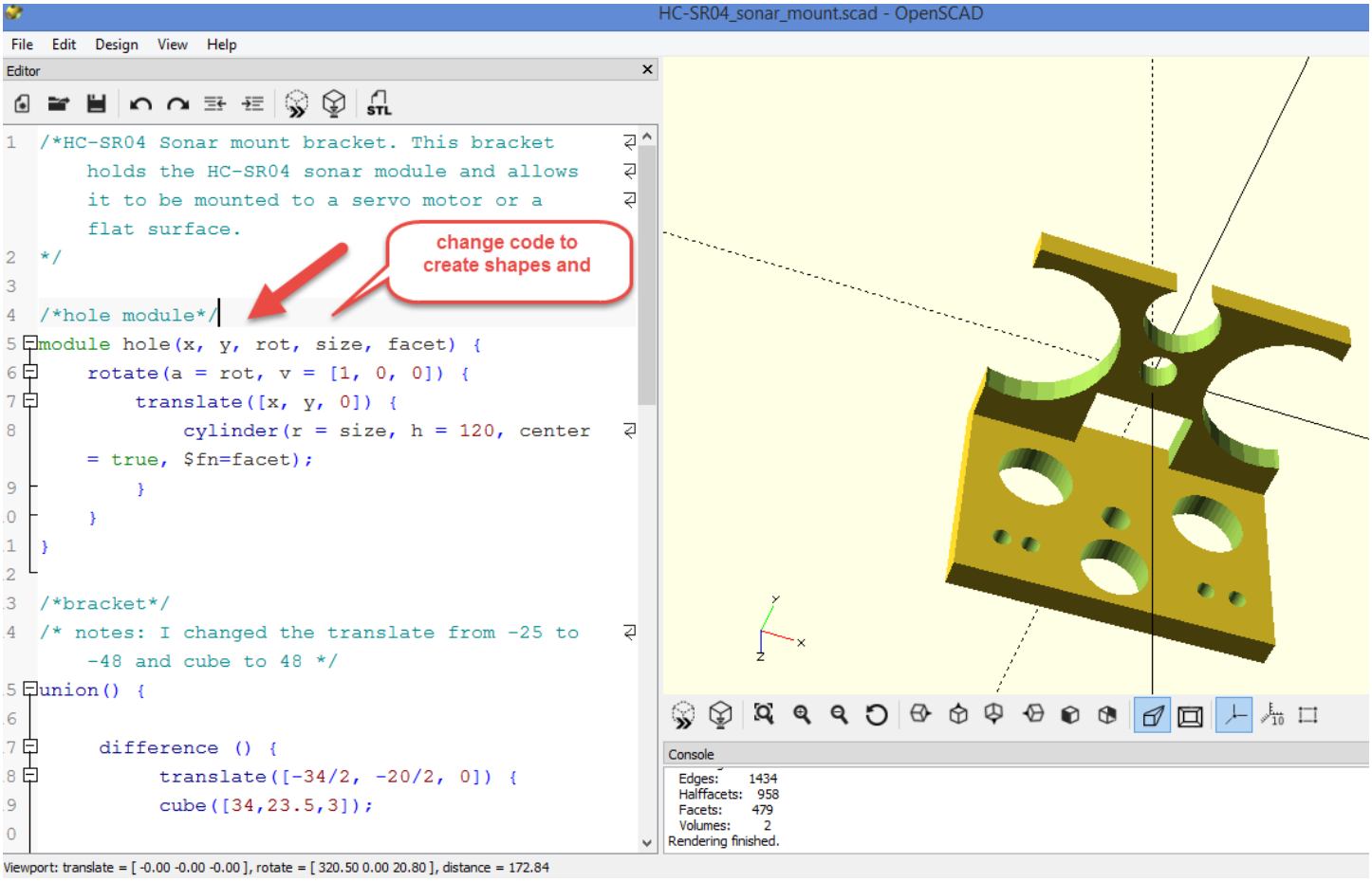
Note: if you use the Makerbot 3D printer you will need to export the STL as .makerbot file

First here is some information on how to design and print your own 3D printed sonar sensor bracket.

STEP 1: Download SCAD (a free solid 3D computer aided drafting modeller you can download at http://www.openscad.org/) or you can find the version we use under Experiment 3 in roboticscity.com

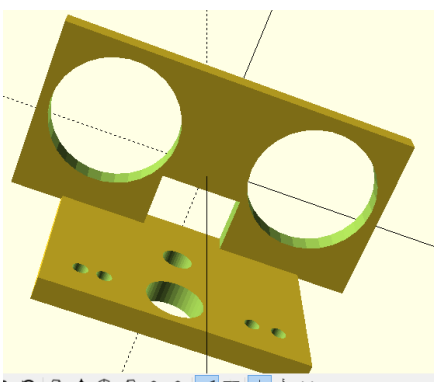
Use the sample files HC-SR04_sonar_mount.scad found on the website http://roboticscity.com to start modifying your bracket.

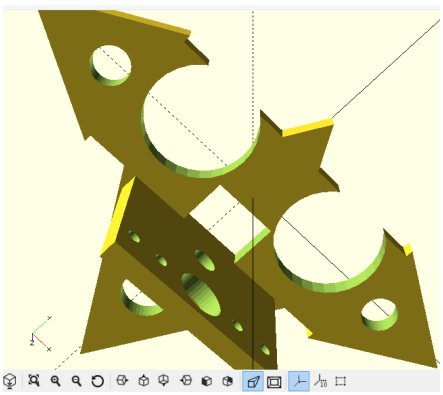
After installing OpenSCAD open the sample HC-SR04_sonar_mount.scad and under Design on the menu select Compile and Render or just Render. This will give you a good idea of what the drawing looks like. On the left hand side you will notice that the code creates the graphics. Start changing the code and experiment with it to create the exact shape you want, but keep the mounting holes on the bottom where they are as you will mount to the small servo motor. Once you are ready to 3D print-it you need to export as STL. Go to File and select Export as STL and save your file with whatever name you want. STL files are files that 3D printers can read and process as machine code.

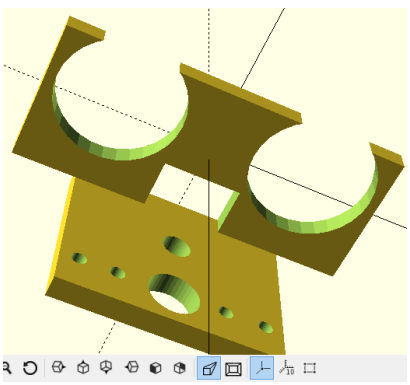



```
/*HC-SR04 Sonar mount bracket. This bracket
    holds the HC-SR04 sonar module and allows
    it to be mounted to a servo motor or a
    flat surface.
*/

/*hole module*/
module hole(x, y, rot, size, facet) {
    rotate(a = rot, v = [1, 0, 0]) {
        translate([x, y, 0]) {
            cylinder(r = size, h = 120, center
    = true, $fn=facet);
        }
    }
}

/*bracket*/
/* notes: I changed the translate from -25 to
    -48 and cube to 48 */
union() {

    difference () {
        translate([-34/2, -20/2, 0]) {
        cube([34,23.5,3]);
```

STEP 2: Open the software on the printer's workstation and be ready to print in 3D.
If using the Makerbot 3D printer open the STL file in the MakerBot software that is on the 3D printer's workstation and export as .makerbot file and send it to the 3D printer by selecting Print It or save it to a thumb drive for later printing. Make sure you select the setting No Raft and the middle setting for resolution before sending the job to print. You can even go lower resolution for faster printing.The Makerbot software does something called slicing and it prepares the file for 3D printing and that is when it saves it as a .makerbot file. It can take up a half hour to print this sensor bracket. For other 3D printers follow the instructions on those printers. Note how SCAD works by creating cubes and holes then you expand the cube or hole and you move it left to right or up and down. Samples shown below.



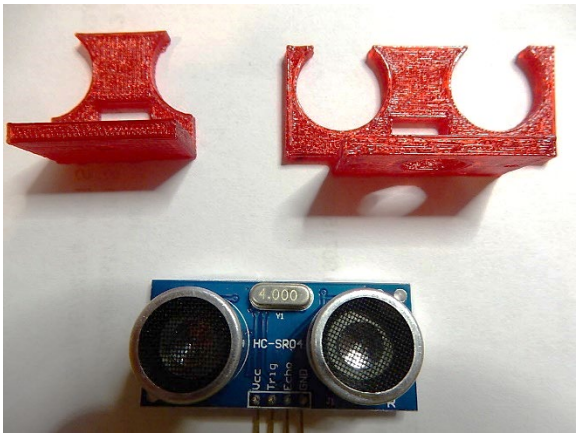


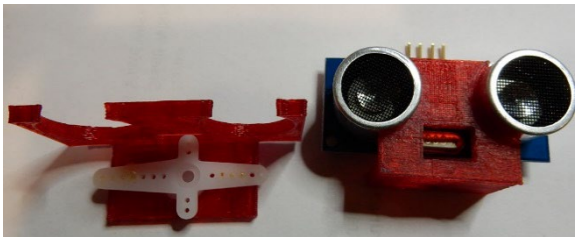


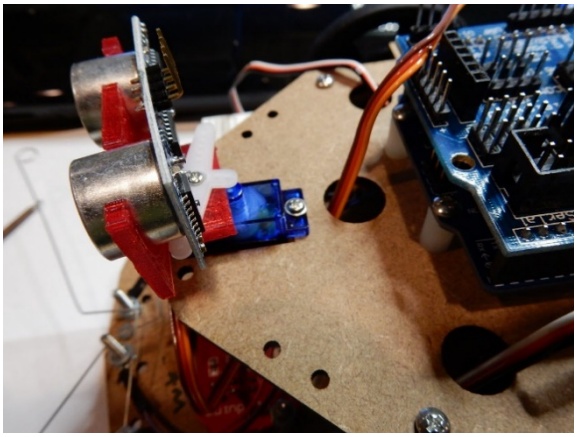
Using a Sonar Sensor

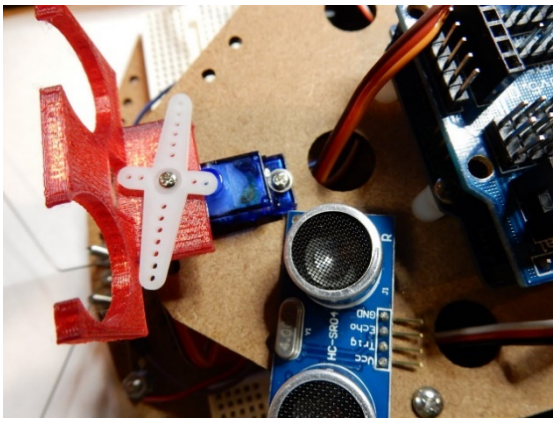
Sonar sensors send and detect a sound wave at a particular frequency that is generated by the Arduino. If you already have your bracket then go ahead and mount it on the robot. Check your sensor connections or you will burn it! GND goes to ground, VCC goes to 5 Volt, Trigger goes to pin 5 and Echo goes to pin 6.

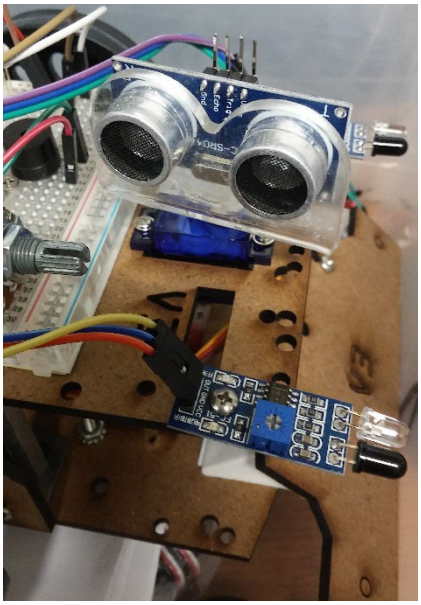
Mount the sensor on the bracket.



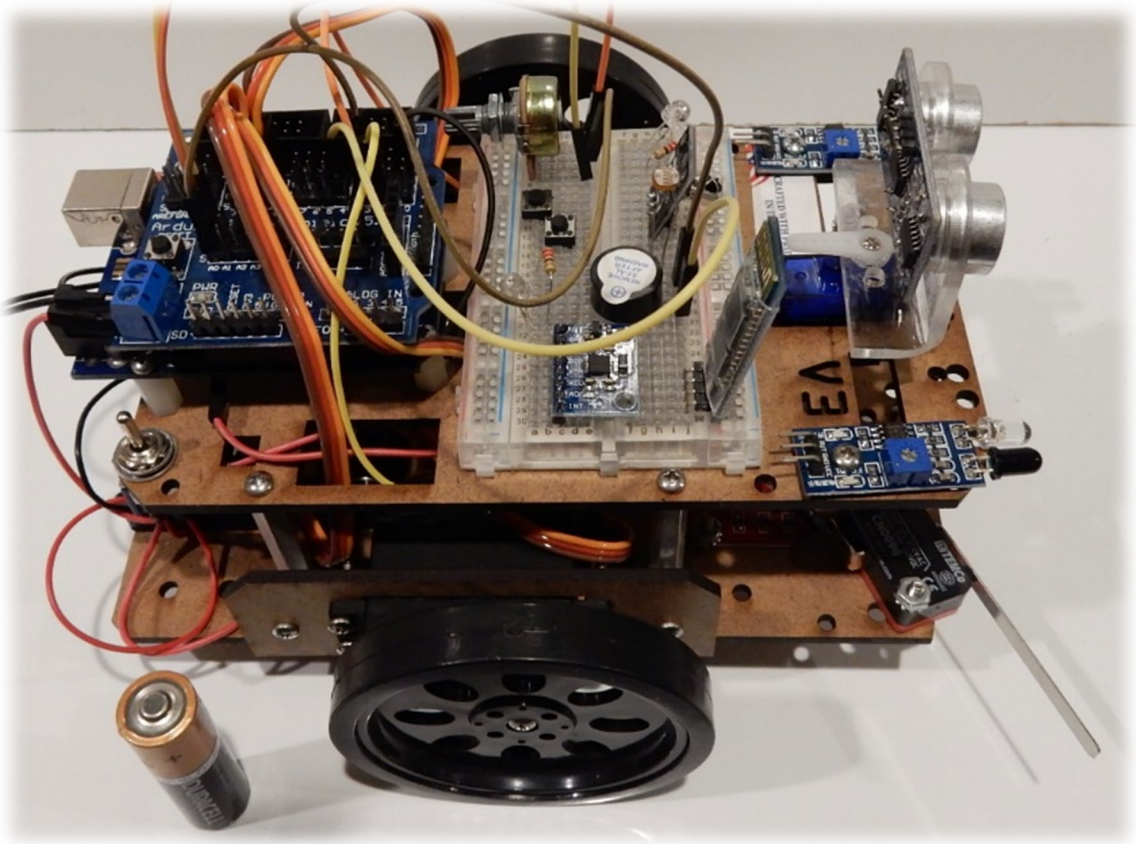








Sensor mounted on robot



Sonar sensors emit an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400

cm or 1” to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module

Features

»Power Supply :+5V DC

»Quiescent Current : <2mA

»Working Current: 15mA

»Effectual Angle: <15°

»Ranging Distance: 2cm – 400 cm/1” – 13ft

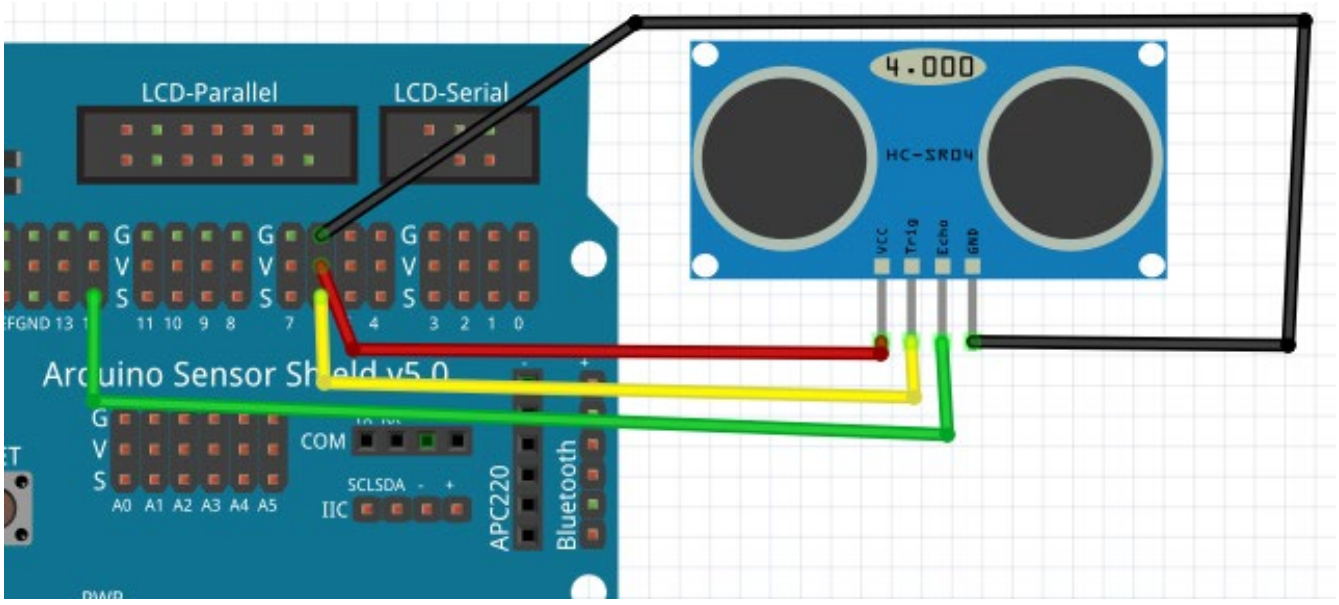
»Resolution: 0.3 cm

»Measuring Angle: 30 degree

»Trigger Input Pulse width: 10uS

»Dimension: 45mm x 20mm x 15mm

Connecting the Sensor



VCC =V, GND=G, TRIG=S(pin 6) and ECHO=S(pin12)

You can download all of the sample code in this experiment under Experiment 3 so there is no need to copy and paste all of this code. Just match the program name to the file. This program does not use a library.

```
// Name: SonarSimple1.ino
// Date Modified: 10/20/16
// This example does not use a library
/*   Ultrasonic sensor Pins:
       VCC: +5VDC
       Trig : Trigger (INPUT) – Pin 6
       Echo: Echo (OUTPUT) - Pin 12
       GND: GND
 */

int trigPin = 6;    //Trig - green Jumper
int echoPin = 12;    //Echo - yellow Jumper
long duration, cm, inches; //define variables on a single line (pretty neat!)

void setup() {
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  cm = (duration/2) / 29.1;
  inches = (duration/2) / 74;

  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(250);
}
```

Open the Serial Monitor to see the sensor output

Using the Newping library

Download from the roboticscity.com Experiment 3 page and expand the NewPing library ZIP file into the Arduino Library folder. You will notice that the code has been reduced significantly.

```
// Name: NewpingSimpleSample1.ino
// Date Modified: 10/20/16
#include <NewPing.h>
#define TRIGGER_PIN 6
#define ECHO_PIN 12
#define MAX_DISTANCE 200 // in CM
// This program tests your sonar sensor by using the newping library. You can see results on serial Monitor
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // se max distance to sense

void setup() {
Serial.begin(9600);
}
void loop() {
delay(50);
unsigned int uS = sonar.ping_cm();
Serial.print(uS);
Serial.println("cm");
}
```

Another simplified sonar sensor tester program. It does not use a library.

```
// Name: SonarSimple2.ino
// Date Modified: 10/20/16
// This code will allow you to test the sonar sensor by displaying distance between an object and sensor
// on the Serial Monitor
int TrigPin = 6;
int EchoPin = 12;
void setup()
{
Serial.begin(9600);
pinMode(TrigPin,OUTPUT);
pinMode(EchoPin,INPUT);
}
void loop()
{
int distance,duration;
digitalWrite(TrigPin,HIGH);    //TrigPin prepare high of more than 10us
delayMicroseconds(11);
digitalWrite(TrigPin,LOW);
duration = pulseIn(EchoPin, HIGH);
```

```
//EchoPin received high start counting until the receiver to the low, return to the count value
duration = duration/29/2;     //Calculating the distance cm
// The speed of sound is 340 m/s or 29 microseconds per centimeter.
Serial.print(duration);   //Serial display output
//distance
Serial.print("cm");
Serial.println();
delay(1000);
}
```

Simple Test Sonar Sensor Program and uses the pulseIn ( ) function

```
// Name: SonarSimple3.ino
// Date Modified: 10/20/16

const int trigPin = 6;
const int echoPin = 12;

// defines variables
long duration;
int distance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}

void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance= duration*0.034/2;

// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
```

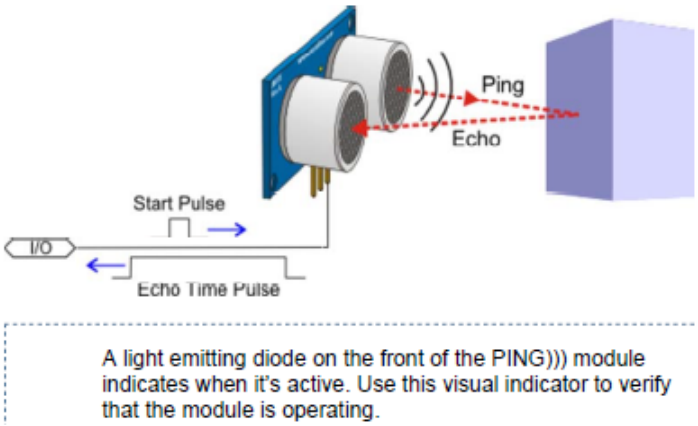
```
}
```

In the loop first you have to make sure that the trigPin is clear so you have to set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 µs. Using the pulseIn() function you have to read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the pulsIn() function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously. At the end we will print the value of the distance on the Serial Monitor.

Specs from manufacturer.

****

1: Voltage: DC+5V
2: Static current: less than 2mA
3: Output Level: High 5V/ Low 0V
4: Sensor angle: <=15 degrees
5: Detection Range: 2cm~450cm
6: Precision: About 2mm



Easy to Connect mode: Just 4 Pins: VCC, Trig (control side), Echo (receiving end), and GND.
(1) Use I/O TRIG, get a pulse signal which is at least 10us long, this will activate the module to start detecting;
(2) The ultrasonic module will automatically send eight 40khz square waves, automatically detects whether there is a ping signal.
(3) When there is ping signal back, the ECHO I/O will output a high level, the duration of the high-level signal is the time from ultrasonic launch to return. As a result, the Measured distance = (T(Time of High Level output) * (340M / S)) / 2;

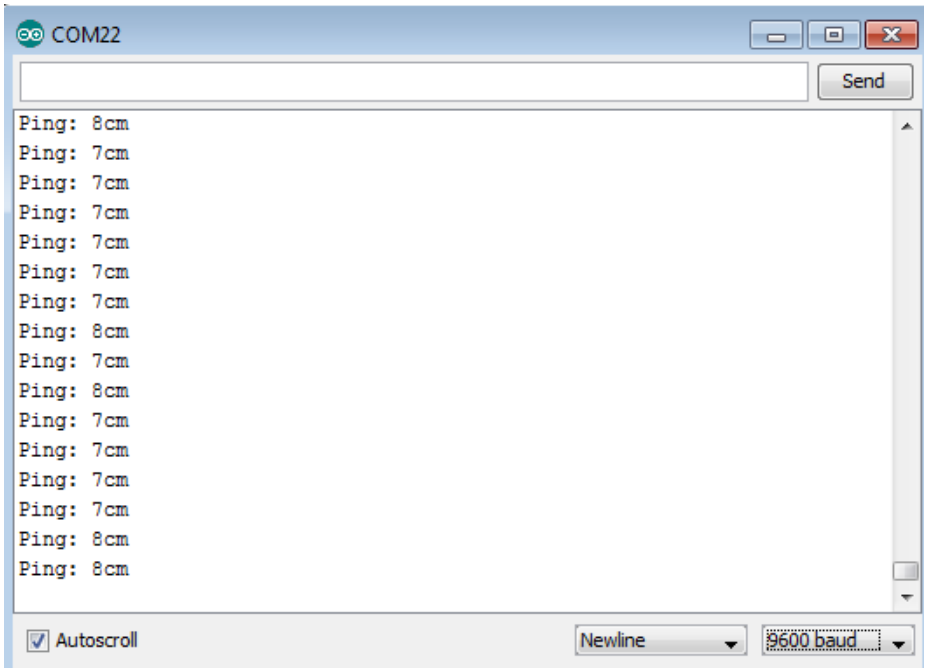
Connection:
VCC: +5V
TRIGER: Send start pulse
Echo: Rturned Pulse
GND: GND

Did you know you can use a single Arduino pin to do both Trig and Echo. The NewPing library takes care of the timing between sending and receiving the pulse so this gives you an extra pin to connect something else to the Arduino.
You will need the NewPing.h Library imported.

Remember to copy the NewPing folder you unzipped to the Arduino\Library folder before opening Arduino so that the library gets registered.

First Connect Trigger and Echo together to signal pin 6 of the Arduino

```
// Name: NewpingSimpleSample2.ino
// Date Modified: 10/20/16
#include <NewPing.h>
#define PING_PIN 6
// Arduino pin 6 tied to both trigger and echo pins on the ultrasonic sensor.
#define MAX_DISTANCE 200
// Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

NewPing sonar(PING_PIN, PING_PIN, MAX_DISTANCE);
// NewPing setup of pin and maximum distance.

void setup()
{
Serial.begin(9600);
// Open serial monitor at 9600 baud to see ping results.
}

void loop()
{
delay(50);
// Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
unsigned int uS = sonar.ping();
// Send ping, get ping time in microseconds (uS).
Serial.print("Ping: ");
Serial.print(uS / US_ROUNDTRIP_CM);
// Convert ping time to distance and print result (0 = outside set distance range, no ping echo)
Serial.println("cm");
}
```



You should see the values on the serial monitor

The Arduino has certain pins reserved to be used by it three timers. If you use those pins while also using timers then you can have erratic behavior in your program. These are all of the PWM pins

Pins 5 and 6: controlled by Timer0
Pins 9 and 10: controlled by Timer1
Pins 11 and 3: controlled by Timer2

Short tutorial:

http://blog.oscarliang.net/arduino-timer-and-interrupt-tutorial/

Timers and Interrupts in Robotics – great example to try so you can see how interrupts work in the Arduino UNO environment for multitasking. Experiment by adding extra tasks in it. At the end of this instruction manual is a finalized program using this example.

++

Timers and Interrupts so the robot can do more than one thing at once such as read sensor values, move servo turret and move servo motors plus make sound.

```
// Name: NewpingEventTimer.ino
// Date Modified: 10/20/16
// This example shows how to use NewPing's ping_timer method which uses the Timer2 interrupt to get the
// ping time. The advantage of using this method over the standard ping method is that it permits a more
// event-driven sketch which allows you to appear to do two things at once. An example would be to ping
// an ultrasonic sensor for a possible collision while at the same time navigating. This allows a
// properly developed sketch to multitask. Be aware that because the ping timer method uses Timer2,
// other features or libraries that also use Timer2 would be effected. For example, the PWM function on
// pins 3 & 11 on Arduino Uno (pins 9 and 11 on Arduino Mega) and the Tone library. Note, only the PWM
// functionality of the pins is lost (as they use Timer2 to do PWM), the pins are still available to use.
```

```
#include <NewPing.h>

#define TRIGGER_PIN   6 // Arduino pin tied to trigger pin on ping sensor.
#define ECHO_PIN      12 // Arduino pin tied to echo pin on ping sensor.
#define MAX_DISTANCE 200
// Maximum distance we want to ping for (in centimeters).
// Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.

unsigned int pingSpeed = 50;
// How frequently are we going to send out a ping (in milliseconds). 50ms would be 20 times a second.
unsigned long pingTimer;
  // Holds the next ping time.

void setup() {
Serial.begin(9600);
// Open serial monitor at 115200 baud to see ping results.
pingTimer = millis(); // Start now.
}

void loop() {
  // Notice how there's no delays in this sketch to allow you to do other processing in-line while doing distance
pings.
  if (millis() >= pingTimer) {
 // pingSpeed milliseconds since last ping, do another ping.
    pingTimer += pingSpeed;
    // Set the next ping time.
    sonar.ping_timer(echoCheck);
 // Send out the ping, calls "echoCheck" function every 24uS where you can check the ping status.
  }
  // Do other stuff here, really. Think of it as multi-tasking like scanning for objects
}

void echoCheck() {
 // Timer2 interrupt calls this function every 24uS where you can check the ping status.
  // Don't do anything here!
  if (sonar.check_timer()) {
 // This is how you check to see if the ping was received.
 // Here's where you can add code.
    Serial.print("Ping: ");
    Serial.print(sonar.ping_result / US_ROUNDTRIP_CM);
// Ping returned, uS result in ping_result, convert to cm with US_ROUNDTRIP_CM.
    Serial.println("cm");
  }
```

```
  // Don't do anything here!
}
```

Ultrasonic.h Library

Another library and a better one is the Ultrasonic library – mainly because it simplifies certain commands. Make sure you import the Ultrasonic library to Arduino\libraries\

```
// Name: UltrasonicDemo.ino
// Date Modified: 10/20/16
// Show the distance in centimeters and in inches on Serial Monitor

#include "Ultrasonic.h"
Ultrasonic ultrasonic(6,12); //Ultrasonic ultrasonic(Trig,Echo);

void setup() {
Serial.begin(9600);
}

void loop()
{
  Serial.print(ultrasonic.Ranging(CM));
  Serial.print("cm");
  Serial.print("    ");
  Serial.print(ultrasonic.Ranging(INC));
  Serial.println("in");
  delay(100);
}
```

Another timing example using the ultrasonic library

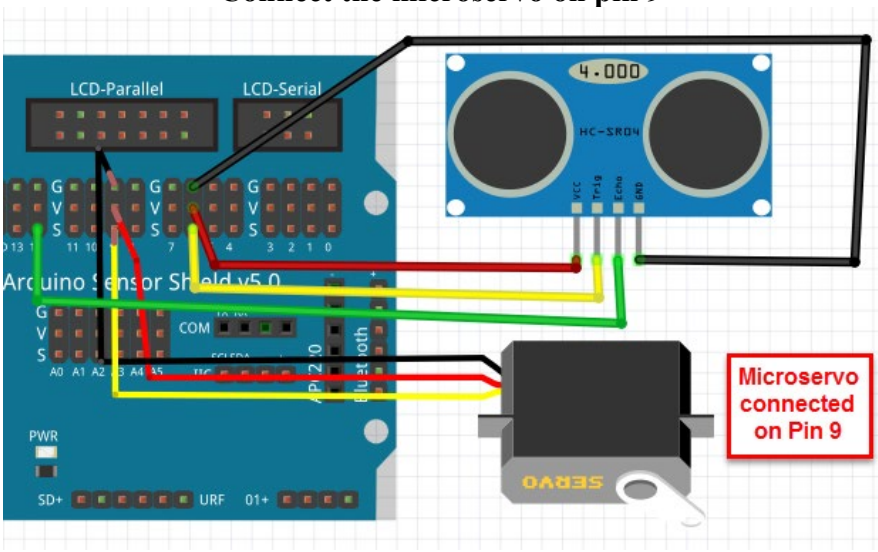
```
// Name: UltraSonicTimerDemo2.ino
// Date Modified: 10/20/16
//Uses the ultrasonic.Timing ( ) function from the ultrasonic library
// Ultrasonic - Library for HR-SC04 Ultrasonic Ranging Module.
//timing function

#include <Ultrasonic.h>

Ultrasonic ultrasonic(9,8); // (Trig PIN,Echo PIN)

void setup() {
  Serial.begin(9600);
}

void loop()
{
  Serial.print(ultrasonic.Timing());
  Serial.println(" ms" ); // milliseconds
```

```
  delay(100);
}
```

Scanning the area and reporting back output angle using the Ultrasonic.h library. Note: this experiment only requires to detect objects in front of it and avoid hitting them, this sample first scans and selects where to go based on what it finds. This example is not a complete program, but it gives you a good idea or a more advanced program, but it does not use Timers or Interrupts so that is why it stops and scans then proceeds rather than scanning and proceeding at the same time. This example should be useful in trying to solve the requirements for this experiment as assigned in the class.

Range Sweep – gives output of angle on serial monitor
The servo motor can be used to move the sensor 180 degrees, while the sensor measures distance at specific angle steps. The range sweep will provide the robot with a view of the distances ahead.

Connect the microservo on pin 9




```
// Name: UltrasonicServoRotate.ino
// Date Modified: 10/20/16
// This code rotates the range sensor 180 degrees and reads distances
// then reports the angle at which distance is maximum - on serial monitor
// with this information you can have a robot navigate
// where the distance is the longer on the monitor

#include "Ultrasonic.h"
#include <Servo.h>
Ultrasonic ultrasonic(6,12); // 6->trig, 12->echo
Servo myservo; // create servo object to control a servo
const int STEP = 5;
const int SIZE = 37;  // Size of distance array =(180/STEP) +1
int dist[SIZE];
int rDelay;
int mid = 90;      // angle of the forward direction

void setup()
```

```
{
Serial.begin(9600);
myservo.attach(9, 570, 2320); // attaches the mini servo on pin 9 to the servo object
rDelay = 10 * STEP;     // assuming 10ms/degree speed
}

void loop()
{
rangeSweep(mid-90, mid+90, dist);
disp(dist);
int angle = getAngle(dist);
Serial.println(angle);
}

/**Reads distance over 180 degrees twice in left-to-right sweep, then right-to-left
sweep and averages the readings  */
void rangeSweep(int st, int en, int dist[])
{
int pos = 0;   // variable to store the servo position
for(pos = st; pos<en; pos+=STEP)
{
myservo.write(pos);              // tell servo to go to position in variable 'pos'
delay(rDelay);     // waits 10ms/degree for the servo to reach the position
dist[int(pos/STEP)] = ultrasonic.Ranging(CM);
}
for(pos = en; pos>st; pos-=STEP)
{
myservo.write(pos);     // tell servo to go to position in variable 'pos'
delay(rDelay);
dist[int(pos/STEP)] += ultrasonic.Ranging(CM);
dist[int(pos/STEP)] /= 2;
}
}

/** Prints the range readings to the serial monitor */
void disp(int dist[])
{
for(int i = 0; i < SIZE; i++)


{
Serial.print(i*STEP);
Serial.print(", ");
Serial.println(dist[i]);

}
}
```

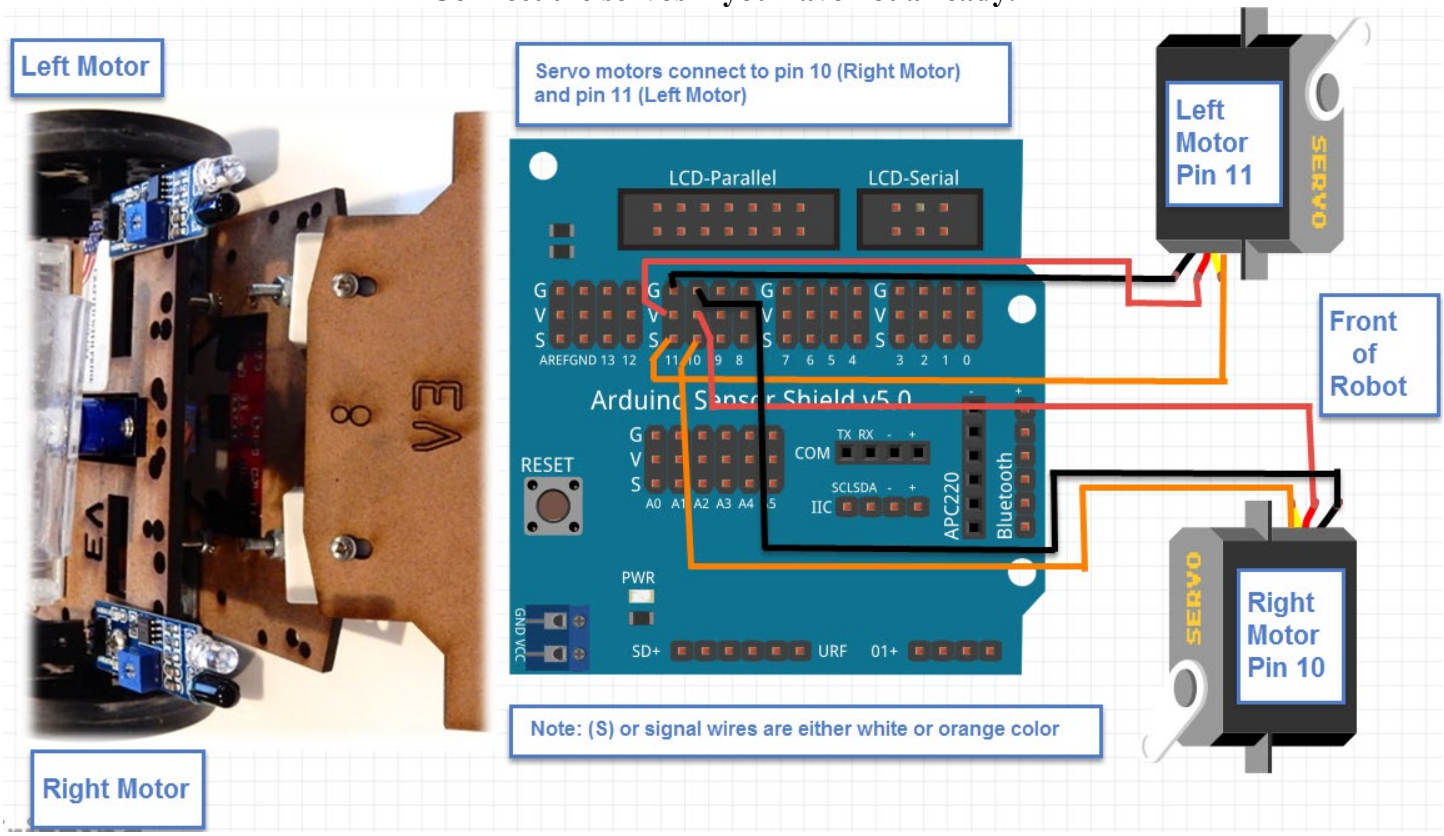
```
/** Get the angle at which the distance is maximum */
int getAngle(int dist[])
{
int maxDist=0;
int angle=mid;
for(int i = 0; i < SIZE; i++)
{
if(maxDist<dist[i]) {
maxDist = dist[i];
angle = i * STEP;
}
}
return angle;
}
```

The code snippet below shows that the sensor is rotated 180 degrees twice in each direction and the readings from each direction are averaged to get a better estimate of the distance. Two important things to mention here: First, some time must be allowed after the write() command before reading the senor to ensure the motor reached the position. The speed of the servo is 5ms/degree but example below allows for 10ms/degree to be sure. Second, the time it takes for ultrasonic.ranging(CM) command to return value depends on the distance of the objects ahead. Therefore, you may notice that while the sensor is rotating, it slows down when there are empty spaces ahead.

```
void rangeSweep(int st, int en, int dist[]) {
  int pos = 0;
  for(pos = st; pos < en; pos+=STEP) {
    myservo.write(pos);
    delay(rDelay);
    dist[int(pos/STEP)] = ultrasonic.Ranging(CM);
  }
  for(pos = en; pos > st; pos-=STEP) {
    myservo.write(pos);
    delay(rDelay);
    dist[int(pos/STEP)] += ultrasonic.Ranging(CM);
    dist[int(pos/STEP)] /= 2;
  }
}
```

Example 4: Simple and tested program to read a ping sensor and make the robot move back when it detects something in front of it. Please note that this one does not include code for the turret (third small servo that scans). Is a good program to use if you don't need to scan, but want to use a sonar sensor for autonomous navigation.

Connect the servos if you have not already.




```
// Name: SonarServosBack.ino
// Date Modified: 10/20/16
// robot moves back when it detects an object infront of it
#include <Servo.h>     // Include servo library
Servo servoLeft;       // Declare left and right servos
Servo servoRight;

int TrigPin=6;         //connect Trigger to pin 5
int EchoPin=12;

void setup() {
//Serial.begin(9600);
pinMode(TrigPin,OUTPUT);
pinMode(EchoPin,INPUT);
servoLeft.attach(11);   // Attach wheel servo
servoRight.attach(10);  // Attach wheel servo
}

void loop()
{
int distance, duration;
digitalWrite(TrigPin, HIGH);
delayMicroseconds(11);
```

```
digitalWrite(EchoPin, LOW);
duration = pulseIn(EchoPin, HIGH);
distance = duration/29/2;

//Serial.print(distance);
//Serial.print("cm");
//Serial.println();
delay(100);

if (distance<10 && distance>0)
{
//backward
servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
servoRight.writeMicroseconds(1700);              // Right wheel counterclockwise
delay(2000);       // ...for 2 seconds

//left
servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
servoRight.writeMicroseconds(1300);              // Right wheel clockwise
delay(600);// ...for 0.6 seconds

//left
servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
servoRight.writeMicroseconds(1300);              // Right wheel clockwise
delay(600);// ...for 0.6 seconds
} else {
//forward
servoLeft.writeMicroseconds(1700);        // Left wheel counterclockwise
servoRight.writeMicroseconds(1300);              // Right wheel clockwise
}
digitalWrite(TrigPin, LOW);
delayMicroseconds(11);
digitalWrite(EchoPin, HIGH);
}
```

Try this Example to make the robot follow you when you put your hand in front of it. Extra sample (optional)

```
// Name: SonarFollowsYou.ino
// Date Modified: 10/20/16
// this program will follow you when you put your hand in front of it
#include <Servo.h> // Include servo library
Servo servoLeft; // Declare left and right servos
Servo servoRight;
int TrigPin=6;
int EchoPin=12;
void setup() {
```

```
Serial.begin(9600);
pinMode(TrigPin,OUTPUT);
pinMode(EchoPin,INPUT);


servoLeft.attach(11); // Attach left motor
servoRight.attach(10); // Attach right motor
}
void loop()
{
int distance, duration;
digitalWrite(TrigPin, HIGH);
delayMicroseconds(11);
digitalWrite(EchoPin, LOW);
duration = pulseIn(EchoPin, HIGH);
distance = duration/29/2;
Serial.print(distance);
Serial.print("cm");
Serial.println();
delay(100);
if (distance<20 && distance>0)
{
//forward
servoLeft.writeMicroseconds(1300); // Left wheel counterclockwise
servoRight.writeMicroseconds(1700); // Right wheel clockwise
}
else
{
//backward
servoLeft.writeMicroseconds(1700); // Left wheel clockwise
servoRight.writeMicroseconds(1300); // Right wheel counterclockwise
delay(500); // ...for 2 seconds

}
digitalWrite(TrigPin, LOW);
delayMicroseconds(11);
digitalWrite(EchoPin, HIGH);
}
```

Scans back and forth between 0-180 degrees smoothly using the micro servo.

```
// Name: SonarTurretTurning.ino
// Date Modified: 10/20/16
//sonar servo turns 0-180 smoothly
#include <Servo.h>
Servo myServo;
```

```
const int delayTime = 800;
const int servoPin = 9;

void setup()
{
myServo.attach(servoPin);
turretCenter();
}

void loop()
{
turretStep(0, 180, 150);
turretStep2(0, 180, 150);
delay(delayTime);
}

void turretStep(int minVal, int maxVal, int stepDelay)
{
for(int i=minVal; i<=maxVal; i+=6)
{
myServo.write(i);
delay(stepDelay);
}
// Read ultrasound here
}

void turretStep2(int minVal, int maxVal, int stepDelay)
{
for(int i=maxVal; i>=minVal; i-=6)
{
myServo.write(i);
delay(stepDelay);
}
// Read ultrasound here
}

void turretCenter()
{
myServo.write(90);
delay(1500);
}
```

It scans between 0-180 degrees using the micro servo, but it also outputs the sonar range values at the same time using Timers.

```
// Name: SonarTurretTurnsValues.ino
// Date Modified: 10/20/16
// This code uses timers to help multitask.
```

```
//You will not see delays added for the pinging
// only for the turret motions from 0 to 180.

#include <NewPing.h>
#include <Servo.h>
#define TRIGGER_PIN 6
#define ECHO_PIN      12
#define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters).
//Maximum sensor distance is rated at 400-500cm.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
unsigned int pingSpeed = 50;
// How frequently are we going to send out a ping (in milliseconds). 50ms would be 20
times a second.
// you can probably experiment with this number for more accuracy
unsigned long pingTimer;

Servo myServo;
const int servoPin = 9;   // turret servo
Servo servoLeft;  // Declare left and right servos
Servo servoRight;

void setup()
{
Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
pingTimer = millis(); // Start now - part of using timers for multitasking
myServo.attach(servoPin);
servoLeft.attach(11);
servoRight.attach(10);
}
void loop()
{
myServo.write(180);
delay(300);
myServo.write(120);
delay(300);
myServo.write(0);
delay(300);
myServo.write(60);
delay(300);
myServo.write(90);
delay(300);
if (sonar.ping_result / US_ROUNDTRIP_CM<70 && sonar.ping_result /
US_ROUNDTRIP_CM>0)
{
//backwards
servoLeft.writeMicroseconds(1700);        // Left wheel counterclockwise
servoRight.writeMicroseconds(1300);       // Right wheel clockwise
```

```
//left
servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
servoRight.writeMicroseconds(1300);       // Right wheel clockwise
delay(600);// ...for 0.6 seconds
//left
servoLeft.writeMicroseconds(1300);        // Left wheel clockwise
servoRight.writeMicroseconds(1300);       // Right wheel clockwise
delay(600);// ...for 0.6 seconds
}
else
{
//forward
servoLeft.writeMicroseconds(1300);        // Left wheel counterclockwise
servoRight.writeMicroseconds(1700);       // Right wheel clockwise
}
// Notice how there's no delays in this sketch to allow you to do other processing in-line
while doing distance pings.
if (millis() >= pingTimer)
{
// pingSpeed milliseconds since last ping, do another ping.
pingTimer += pingSpeed;      // Set the next ping time.
 sonar.ping_timer(echoCheck); // Send out the ping, calls "echoCheck" function every
24uS
// where you can check the ping status.
}

}
void echoCheck() {
  // Timer2 interrupt calls this function every 24uS where you can check the ping status.
// Don't do anything here!
if (sonar.check_timer())
{
// This is how you check to see if the ping was received.
Serial.print("Ping: ");
Serial.print(sonar.ping_result / US_ROUNDTRIP_CM);
// Ping returned, uS result in ping_result, convert to cm with US_ROUNDTRIP_CM.
Serial.println("cm");
}
//  Don't do anything here!
}
```

Experimental code – works erratically, but try it if you like:

```
// Name: SonarScanRead.ino
// Date Modified: 10/20/16
// scans and reads values fast
```

```
#include <NewPing.h>
#define TRIGGER_PIN 6 // Arduino pin tied to trigger pin on ping sensor.
#define ECHO_PIN  12   // Arduino pin tied to echo pin on ping sensor.
#define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters).
//Maximum sensor distance is rated at 400-500cm.
#include <Servo.h>
Servo myservo; // create servo object to control a servo
const int STEP = 5;
const int SIZE = 37;  // Size of distance array =(180/STEP) +1
int dist[SIZE];
int rDelay;
int mid = 90;        // angle of the forward direction
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.
Servo servoLeft;    // Declare left and right servos
Servo servoRight;
unsigned int pingSpeed = 50;
// How frequently are we going to send out a ping (in milliseconds). 50ms would be 20 times a
second.
unsigned long pingTimer; // Holds the next ping time.
void setup() {
Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.
pingTimer = millis(); // Start now.

myservo.attach(9, 570, 2320); // attaches the servo on pin 9 to the servo object
rDelay = 10 * STEP;       // assuming 10ms/degree speed

servoLeft.attach(11);      // Attach left signal to P13
servoRight.attach(10);     // Attach right signal to P12
}
void loop()
{
{
rangeSweep(mid-90, mid+90, dist);
int angle = getAngle(dist);


}
if (sonar.ping_result / US_ROUNDTRIP_CM<40 && sonar.ping_result / US_ROUNDTRIP_CM>0)
{
//backwards
servoLeft.writeMicroseconds(1700);    // Left wheel counterclockwise
servoRight.writeMicroseconds(1300);  // Right wheel clockwise
//left
servoLeft.writeMicroseconds(1300);    // Left wheel clockwise
servoRight.writeMicroseconds(1300);  // Right wheel clockwise
delay(600);  // ...for 0.6 seconds
//left
servoLeft.writeMicroseconds(1300);    // Left wheel clockwise
servoRight.writeMicroseconds(1300);  // Right wheel clockwise
delay(600);  // ...for 0.6 seconds
}
```

```
else
{
//forward
servoLeft.writeMicroseconds(1300);    // Left wheel counterclockwise
servoRight.writeMicroseconds(1700);  // Right wheel clockwise
}
// Notice how there's no delays in this sketch to allow you to do other processing in-line while
doing distance pings.
if (millis() >= pingTimer)
{
// pingSpeed milliseconds since last ping, do another ping.
pingTimer += pingSpeed;         // Set the next ping time.
sonar.ping_timer(echoCheck);
// Send out the ping, calls "echoCheck" function every 24uS where you can check the ping
status.
}
// Do other stuff here, really. Think of it as multi-tasking like scanning for obstacles.
}

void rangeSweep(int st, int en, int dist[])
{
int pos = 0;   // variable to store the servo position
for(pos = st; pos<en; pos+=STEP) {
myservo.write(pos);        // tell servo to go to position in variable 'pos'
delay(8);     // waits 10ms/degree for the servo to reach the position
dist[int(pos/STEP)] = (sonar.ping_result / US_ROUNDTRIP_CM);
}
for(pos = en; pos>st; pos-=STEP) {
myservo.write(pos);        // tell servo to go to position in variable 'pos'
delay(8);
dist[int(pos/STEP)] += (sonar.ping_result / US_ROUNDTRIP_CM);
dist[int(pos/STEP)] /= 2;
}
}
/** Get the angle at which the distance is maximum */
int getAngle(int dist[])
{
int maxDist=0;
int angle=mid;


for(int i = 0; i < SIZE; i++)
{
if(maxDist<dist[i])
{
maxDist = dist[i];
angle = i * STEP;
}
}
angle;
}
```

```
void echoCheck()
{
// Timer2 interrupt calls this function every 24uS where you can check the ping status.
// Don't do anything here!
if (sonar.check_timer())
{
// This is how you check to see if the ping was received.
// Here's where you can add code.
Serial.print("Ping: ");
Serial.print(sonar.ping_result / US_ROUNDTRIP_CM);
// Ping returned, uS result in ping_result, convert to cm with US_ROUNDTRIP_CM.
Serial.println("cm");
}
// Don't do anything here!
}
```

I hope this experiment provided you with some understanding on Timers/Interrupts and sonar sensor programming and libraries. Using the examples and this experimental code below create a program that allows the robot to scan 180 first, take readings of the environment where it will collect the distances between the sonar sensor and any object in front of the robot then have the robot navigate to the area with the least obstruction.

Extras

```
// Name: TimerExample.ino
// Date Modified: 10/20/16

// While the NewPing library's primary goal is to interface with ultrasonic sensors, interfacing with
// the Timer2 interrupt was a result of creating an interrupt-based ping method. Since these Timer2
// interrupt methods were built, the library may as well provide the functionality to use these methods
// in your sketches. This shows how simple it is (no ultrasonic sensor required). Keep in mind that
// these methods use Timer2, as does NewPing's ping_timer method for using ultrasonic sensors. You
// can't use ping_timer at the same time you're using timer_ms or timer_us as all use the same timer.

#include <NewPing.h>

#define LED_PIN 13 // Pin with LED attached.

void setup() {
  pinMode(LED_PIN, OUTPUT);
  NewPing::timer_ms(500, toggleLED); // Create a Timer2 interrupt that calls toggleLED in your sketch once
every 500 milliseconds.
}

void loop() {
  // Do anything here, the Timer2 interrupt will take care of the flashing LED without your intervention.
}

void toggleLED() {
  digitalWrite(LED_PIN, !digitalRead(LED_PIN)); // Toggle the LED.
```

}

Final working code

```
// Lab 3 - Navigate with Sonar Final Program  v.10.26.17
//Robot navigate with 180 moving turret using the sonar sensor
//In some cases you migh need to swap motor pins or change angles on code to improve the movements
//code is available to download at roboticscity.com

#include <Servo.h>
#include <NewPing.h>
#define TRIGGER_PIN 6
#define ECHO_PIN 12
#define MAX_DISTANCE 100

// Set global variables
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

Servo sonarServo;
Servo leftServo;
Servo rightServo;

unsigned int pingSpeed = 25;
unsigned long pingTimer; // Holds the next ping time.
const int delayTime = 800;


// Set pins
byte LeftServoPin = 11; // typical pins we use on servos for all labs
byte RightServoPin = 10;
byte SonarServoPin = 9; // turret small servo

void setup()
{
  Serial.begin(9600); // Open serial monitor at 9600 baud to see ping results.

  pingTimer = millis(); // Start now - part of using timers for multitasking
  sonarServo.attach(SonarServoPin);
  leftServo.attach(LeftServoPin);
  rightServo.attach(RightServoPin);

  stop();
  turretCenter();
}
```

```
/*********************************************************
 * Main Loop
 *********************************************************/
void loop()
{
  turretClockwise(30, 150, 100);
  turretCounterClockwise(30, 150, 100);

}
/*********************************************************
 * Sonar Functions
 *********************************************************/

void checkSonar(int direction)
{
  if (sonar.ping_result / US_ROUNDTRIP_CM < 30 && sonar.ping_result / US_ROUNDTRIP_CM > 0)
  {
    back(); //backwards
    if (direction >90)
    {
      leftTurn();
    }
    else
    {
      rightTurn();
    }
  }
  else
  {
    forward ();//forward
  }

  if (millis() >= pingTimer)
  {
    // pingSpeed milliseconds since last ping, do another ping.
    pingTimer += pingSpeed; // Set the next ping time.
   // sonar.ping_timer(echoCheck);
  }
}

// you can uncommment the function below only to test sensor and read values on screen
// might need to uncomment sonar.ping_timer(echoCheck)
/*  void echoCheck()
{ // Timer2 interrupt calls this function every 24uS where you can check the ping status.

  // Don't do anything here!
```

```
  if (sonar.check_timer())
  {
    // This is how you check to see if the ping was received.
    Serial.print("Ping: ");
    Serial.print(sonar.ping_result / US_ROUNDTRIP_CM);
    // Ping returned, uS result in ping_result, convert to cm with US_ROUNDTRIP_CM.
    Serial.println("cm");
  }
  // Don't do anything here!
}

*/

/*********************************************************************************
 * Turret Control (turrentClockwise, turrentCounterClockwise, turretCenter)
 *********************************************************************************/
void turretClockwise(int minVal, int maxVal, int stepDelay)
{
  for(int i = minVal; i <= maxVal; i += 6)
  {
    sonarServo.write(i);
    checkSonar(i);
    delay(stepDelay);
  }
}

void turretCounterClockwise(int minVal, int maxVal, int stepDelay)
{
  for(int  i = maxVal; i >= minVal; i -= 6)
  {
    sonarServo.write(i);
    checkSonar(i); // Read ultrasound here
    delay(stepDelay);
    }
}

void turretCenter()
{
  sonarServo.write(90);
  delay(1500);
}


/*********************************************************************************
 * Movement (forward, leftTurn, rightTurn, back, stop)change angle value for speed
 *********************************************************************************/
void forward() // Forward function - No Delay
```

```
{
  Serial.println(" Moving forward" );
  leftServo.write(105); // Left wheel counterclockwise
  rightServo.write(75); // Right wheel clockwise
}

void forward(int time) // Forward function
{
  Serial.println(" Moving forward" );
  leftServo.write(120); // Left wheel counterclockwise
  rightServo.write(60); // Right wheel clockwise
  delay(time); // Maneuver for time ms
}

void leftTurn()
{
  Serial.println(" Turning left" );
  leftServo.write(75);
  rightServo.write(75);
}

void leftTurn(int time)
{
  Serial.println(" Turning left" );
  leftServo.write(0);
  rightServo.write(0);
  delay(time);
}

void rightTurn()
{
  leftServo.write(105);
  rightServo.write(105);
}

void rightTurn(int time)
{
  leftServo.write(180);
  rightServo.write(180);
  delay(time);
}

void back()
{
  leftServo.write(10);
  rightServo.write(180);
}
```

```
void back(int time)
{
  leftServo.write(10);
  rightServo.write(180);
  delay(time);
}

void stop()
{
  leftServo.write(90);
  rightServo.write(90);
}
```

End Experiment 3