An Ultrasonic 3D scanner

Prithvijit Chakrabarty (prithvichakra@gmail.com)
Kartik Lovekar (kslovekar@gmail.com)

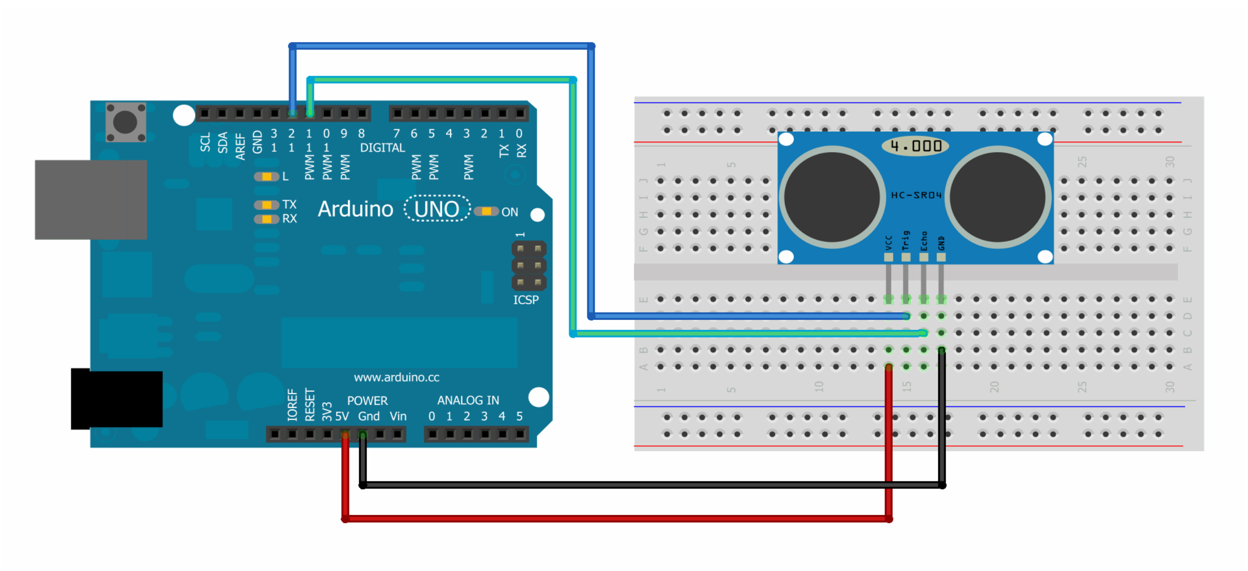
This describes a 3D scanner that uses an ultrasonic sensor to map nearby surfaces.

Components required:

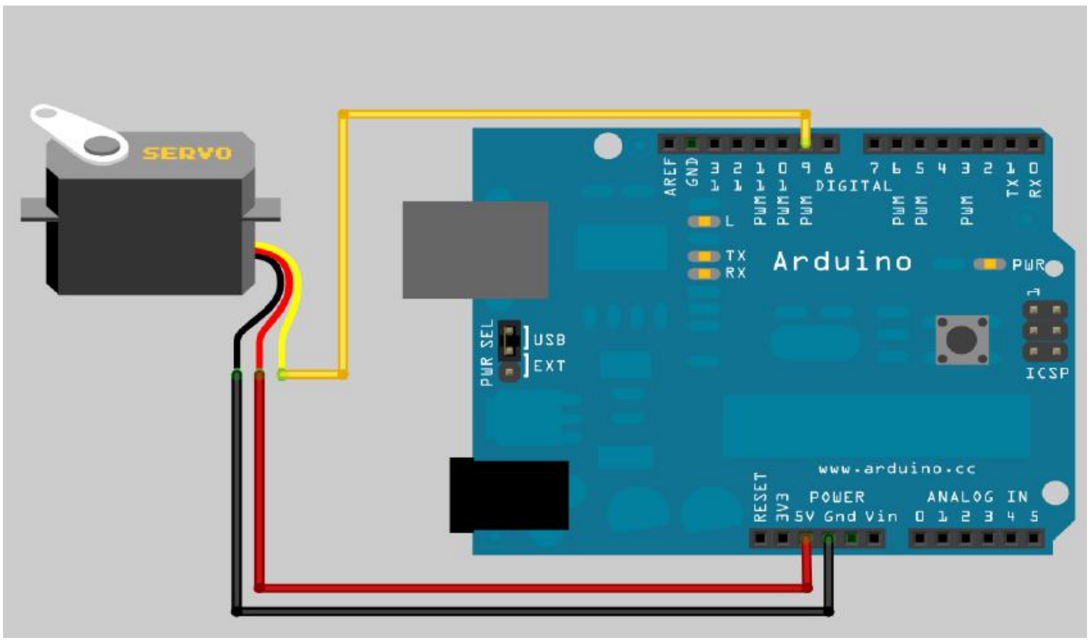
- Arduino uno
- HC-SR04 Ultrasonic sensor
- TowerPro SG2 servos
- Jumper wires
- Bread board

Circuit diagrams:

HC-SR04 with arduino uno

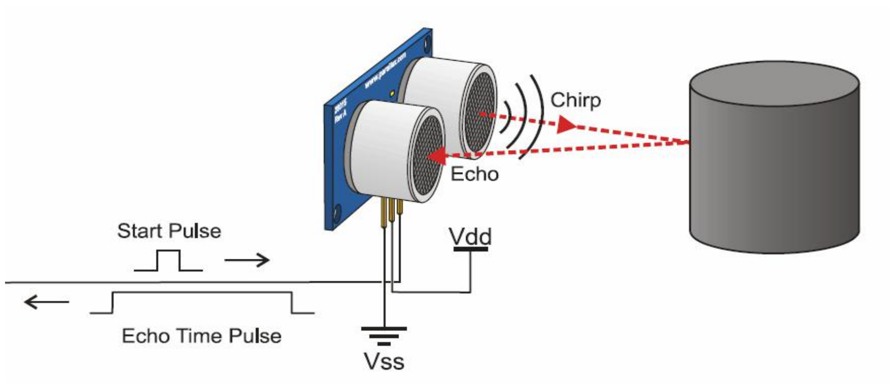


Servo with arduino

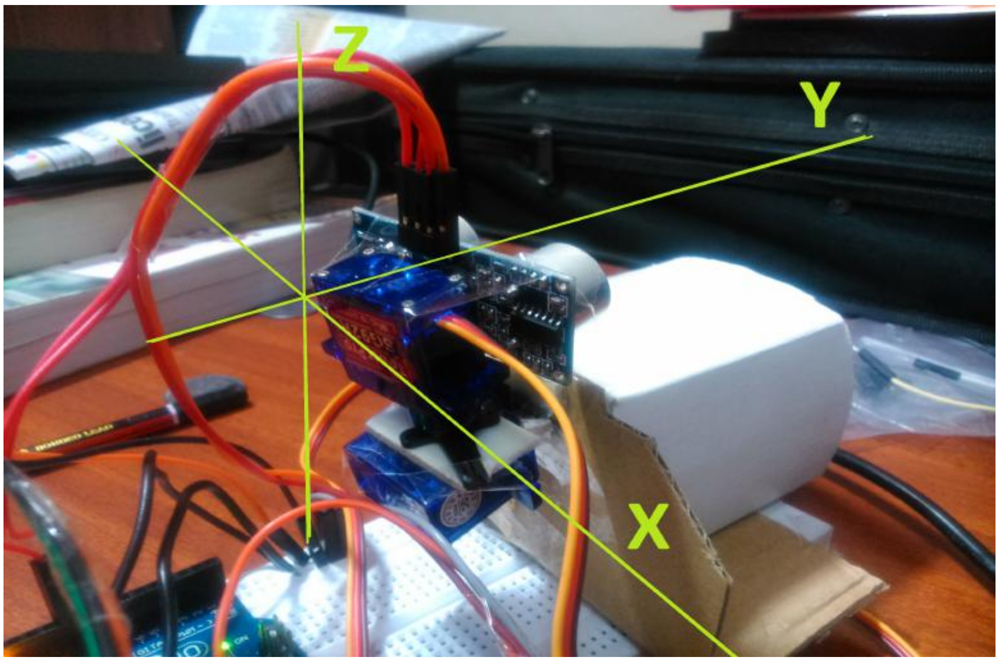


Working:

- Measuring distance:
  The HC-SR04 sensor emits an ultrasonic wave and measures the time taken for the echo to return to approximate the distance to the nearest surface along a given line.



  We control the ultrasonic wave generation by sending a pulse to the pin connected to the “Trigger” pin of the sensore board. On receiving the echo, the sensor sends an echo time pulse to the arduino (through the “Echo” terminal of the sensor board). The length of this pulse is proportional to the distance to the surface.
  The HC-SR04 generates a conical wave, but uses a linear receiver to detect the echo. This causes the sensor to incorrectly detect distances in case the surface is not perpendicular to the face of the sensor board. To resolve this issue, we have used the NewPing library for Arduino, which has built in methods for error correction.
- Rotating the sensor in 3 dimensions:
  The position of the sensor is controlled by two servos (one mounted on top of another). The base servo is attached to a vertical cardboard surface. To get a large surface area to mount the second servo (and also the sensor on it), we taped the blade of the servo instead of its body on the supporting surface.
  The base servo has its body parallel to the ground (on the X-Y plane). On rotation, its base rotates about the Y axis. The second servo has its blade taped on to the body of the base servo. The body of the second servo is perpendicular to that of the first (on the X-Z plane). This rotates about the Z axis.



The rotation of the second servo (the servo on top) gives the value of the angle α. The angle of rotation of the base is β.
The servos are controlled by writing an angle to the analog pin connected to the servo. As we are controlling the servo with Processing, we write the value of the angle (in degrees) on to the port for serial communication with the Arduino. The serial port limits the driver program to read one character at a time. To handle this, we use a protocol to delimit different values:

- Controller to sensor:

<angle>x:	Rotate the base servo by <angle> degrees
<angle>y:	Rotate the mounted servo by <angle> degrees

- Sensor to controller:

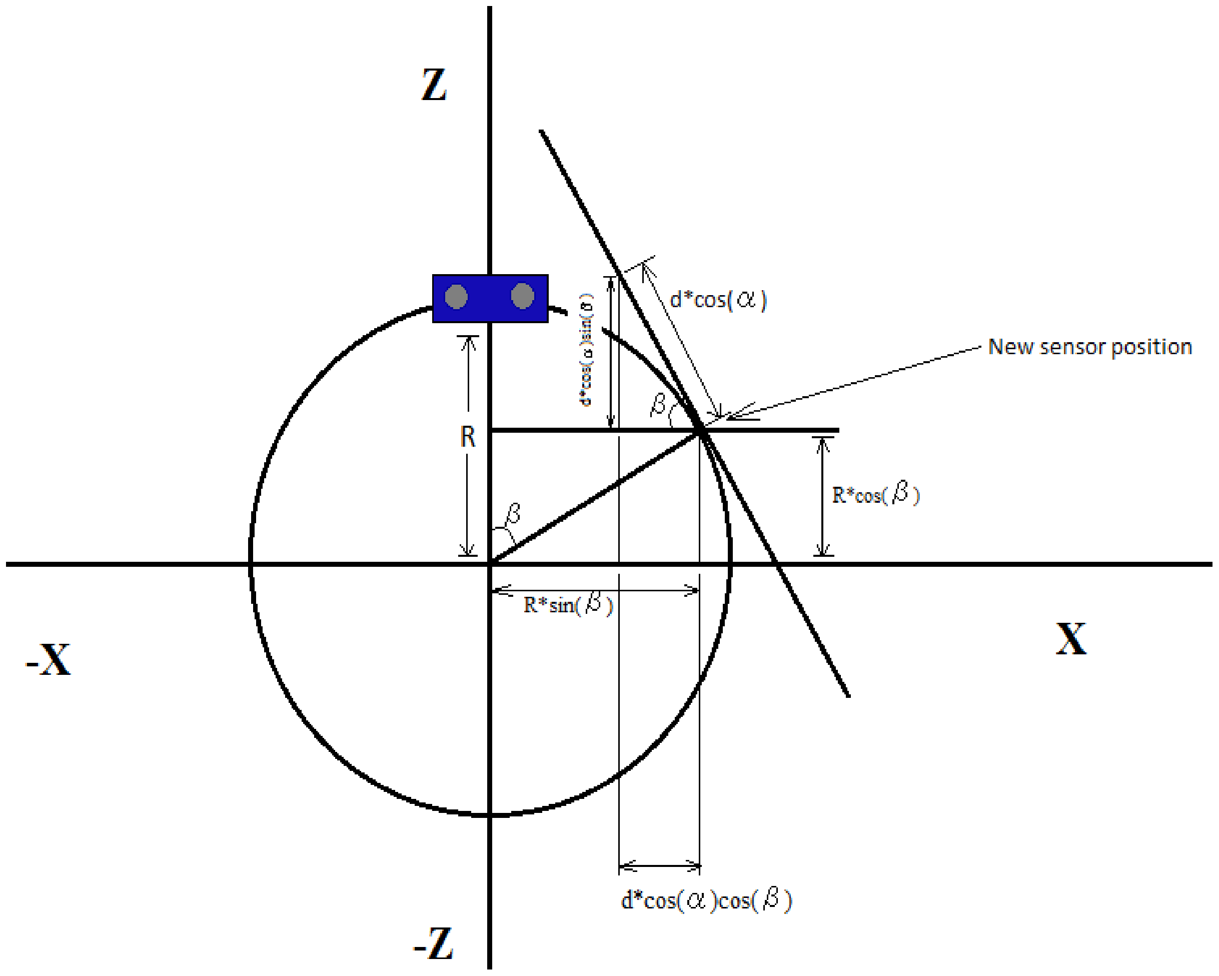
<servo_id>-<angle>:<dist>c:	The sensor detected a distance <dist> when the servo with ID <servo_id> was inclined at angle <angle>.

Writing the functions to read the communication buffer between Processing and the sensor caused synchronization problems. To solve these, we overrode the serialEvent(Serial port) method in Processing. This method is evoked whenever there are characters to be read from the buffer and runs in the background, while the main Processing code controls the servos.

- The servo rotates and a delay is used to ensure that the ultrasonic sensor detects the distance accurately. The Processing code can be modified to control the angular increment per iteration. In every iteration, which constitutes a “sweep” of the sensor, the upper servo rotates from $\alpha = 0$ to $\alpha = 180$ and back, while the base rotates from $\beta = 0$ to $\beta = 70$. The reverse sweep is used to eliminate the stray points (the sensor might produce outlying points at very large distances from the origin). The lower of the two values obtained is used in the 3D point cloud.

Plotting:

- Once both servos completed the whole scan, the detected distances were used to find the cartesian coordinates of the scanned points.
- We used the Processing library ToxicLibs to generate and manipulate the 3D point cloud. The library PeasyCam was used for the rendering and controlled rotation of the 3D space on screen.
- Calculations:



The final expression for the cartesian coordinates are:

- X: $R*\sin(\beta) + d*\cos(\alpha)\cos(\beta)$
- Y: $d*\sin(\alpha)$
- Z: $R*\cos(\beta) + d*\cos(\alpha)\sin(\beta)$

These transformations assume that the body of the base servo is perfectly along the X-Y plane. However, we were unable to position the servo in this position due to the jump wires interfering with the rotation. We measured the actual starting configuaration of the two servos with respect to the target coordinate system.

- Error in $\alpha = 20^o$
- Error in $\beta = 60^o$

To account for this error, we rotated all the points obtained by the previous formula. This is done by multiplying each point with the rotation matrix.

About the X-axis:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

About the Y-axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

About the Z-axis:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The above transformations reduce to a set of 2D transformations which we use by eliminating the third variable.
Thus,the correction matrices were:

- Base sevo: $\begin{bmatrix} \cos 60 & -\sin 60 \\ \sin 60 & \cos 60 \end{bmatrix}$

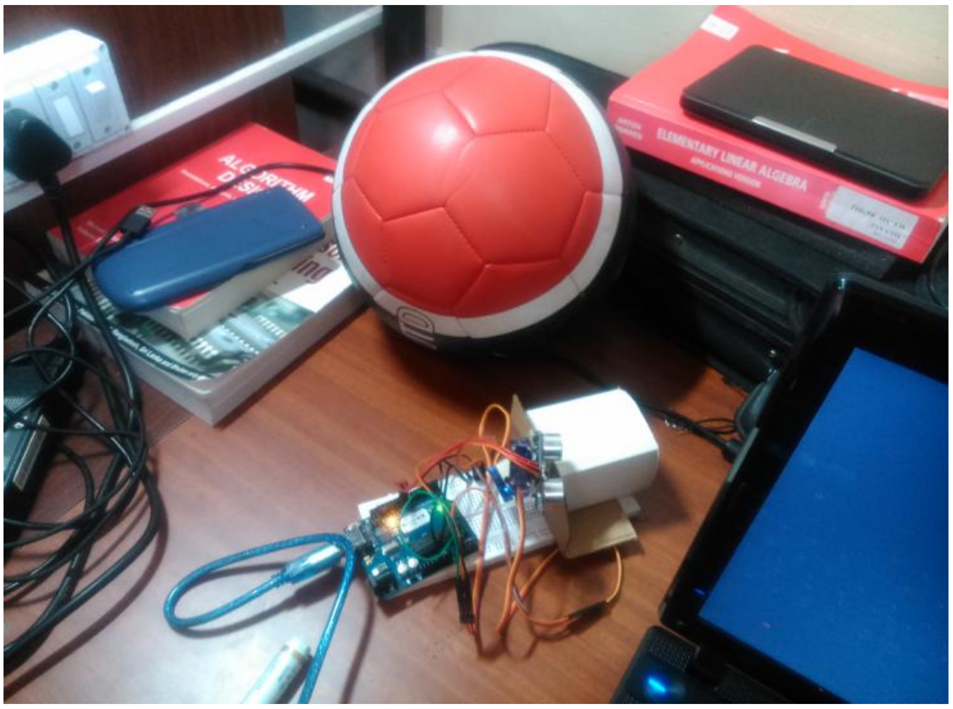
- Mounted servo: $\begin{bmatrix} \cos 20 & -\sin 20 \\ \sin 20 & \cos 20 \end{bmatrix}$

The first transformation is applied on the X and Z coordinates and the second is used on the X and Y coordinates.

Screenshots:

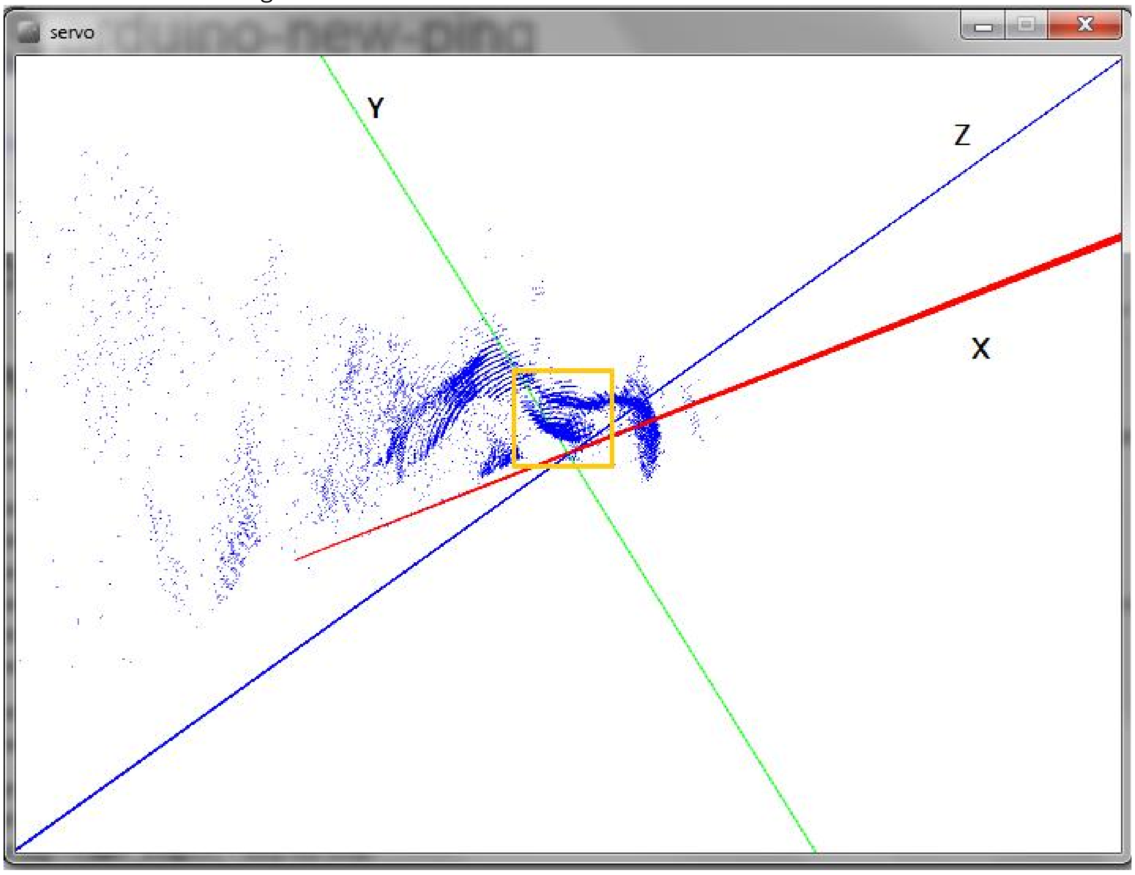
We have uploaded the complete source code on to GitHub. It can be viewed here: https://gist.github.com/PCJohn/fa94b020d8710cabe29c .

Scan setup:



Scan results (1⁰ increment per servo):

The yellow box is the scan region for the football



Scan setup:

Scan results (3^{o} increment per servo):
The yellow box encloses the scan region for the newspaper. There is a noticeable triangular depression in the point cloud of the newspaper's surface, which shows us what the paper looks like when seen from above (through the Z axis).