Lebanese university
Faculty of engineering

3D Imaging using Sound Waves

Presented to: Dr. Houda Ghamloush

Prepared By:
Ahmad N. Ismail
Ahmad H. Ismail

Contents

List of Figures

Chapter 1

General Introduction

Distance measurement using sound echo has multiple purposes. It's used on some terrestrial vehicles to help avoid collisions and assist in parking. The concept was further extended to capture detailed 3D images in conditions where regular cameras are not viable. It is used by vessels and submarines to detect underwater objects, thus playing a critical role in navigation[1]. The concept is also used medically as a safe alternative to X ray imaging. Our objective is to augment a simple distance sensor and use it to take 3D images.

Chapter 2

Materials and methods

2.1 Introduction

Components used:

- Arduino UNO
- Ultrasonic sensor HC-SR04
- Servo motor SG-90
- Stepper motor 28BYJ-48
- 9V/600mA AC to DC adapter
- Breadboard power supply (5V/3.3V)
- Breadboard

2.2 Arduino UNO

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button[2].

2.2.1 Specifications

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA

- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

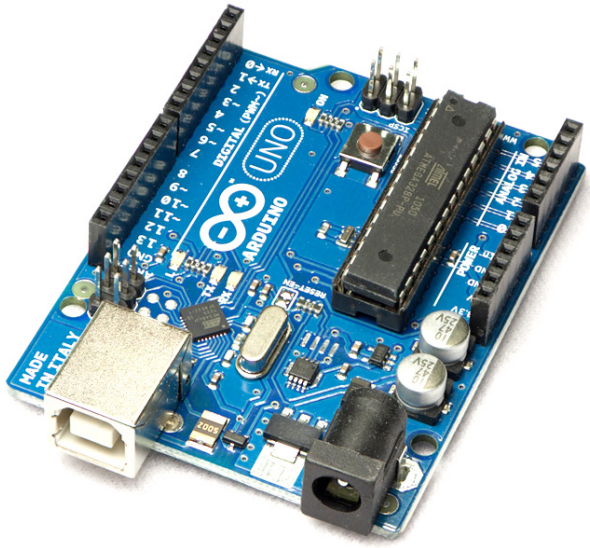

Figure 2.1: Arduino UNO

2.3 Ultrasonic Sensor

The HC-SR04 Ultrasonic Distance Sensor is an inexpensive device that is very useful for robotics and test equipment projects. This tiny sensor is capable of measuring the distance between itself and the nearest solid object. The HC-SR04 can be hooked directly to an Arduino or other microcontroller and it operates on 5 volts. This ultrasonic distance sensor is capable of measuring distances between 2 cm to 400 cm. It's a low current device so it's suitable for battery powered devices.[3]

2.3.1 Specifications

- Operating Voltage: 3.3Vdc 5Vdc
- Quiescent Current: 2mA
- Operating Current: 15mA
- Operating Frequency: 40KHz
- Operating Range & Accuracy: 2cm 400cm (1in 13ft) ± 3mm
- Sensitivity: -65dB min

- Sound Pressure: 112dB
- Effective Angle: 15°
- Connector: 4-pins header with 2.54mm pitch
- Dimension: 45mm x 20mm x 15mm
- Weight: 9g [4]

2.3.2 Mode of operation

1. A 5 volt pulse of at least 10 μS (10 microseconds) in duration is applied to the Trigger pin.
2. The HC-SR04 responds by transmitting a burst of eight pulses at 40 KHz. This 8-pulse pattern makes the "ultrasonic signature" from the device unique, allowing the receiver to discriminate between the transmitted pattern and the ultrasonic background noise.
3. The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes high to start forming the beginning of the echo-back signal. See Figure 2.2
4. If the pulse isn't reflected back then the Echo signal will timeout after 38 ms (milliseconds) and return low. This produces a 38 ms pulse that indicates no obstruction within the range of the sensor.
5. If the pulse is reflected back the Echo pin goes low when the signal is received. This produces a pulse whose width varies between 150 μS to 25 mS, depending upon the time it took for the signal to be received.
6. The width of the received pulse is used to calculate the distance to the reflected object. Remember that the pulse indicates the time it took for the signal to be sent out and reflected back so to get the distance we have to divide the duration in half.[5]

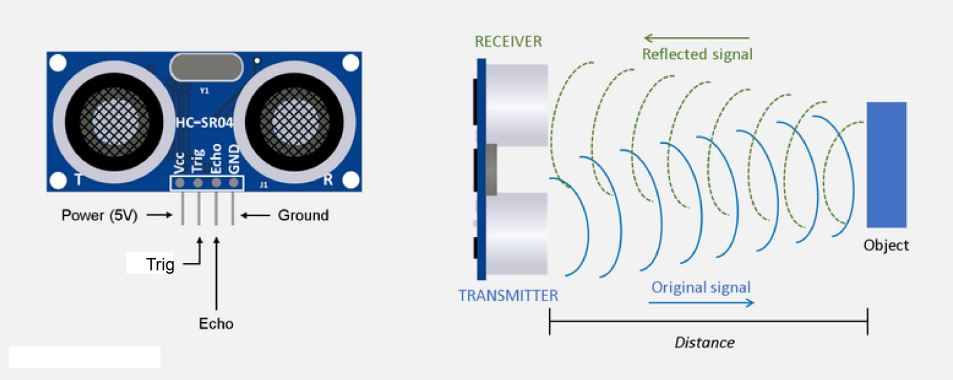

Figure 2.2: Ultrasonic sensor operation

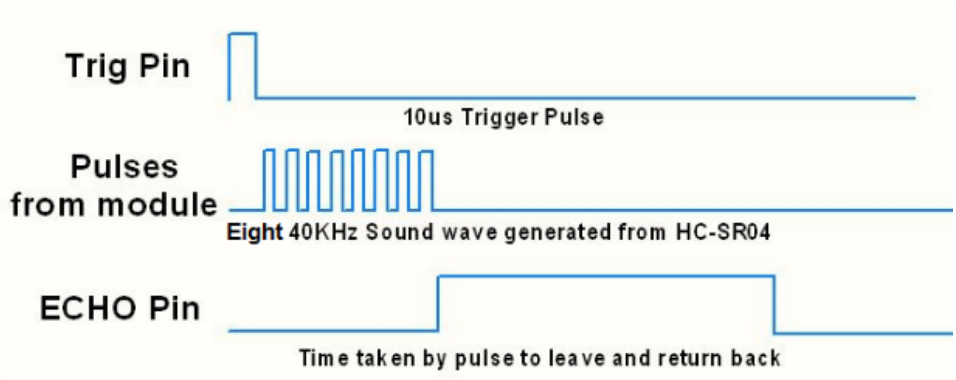

Figure 2.3: Ultrasonic module timing diagram

2.4 Servo Motor

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback.

2.4.1 Wire Configuration

Wire color	Description
Brown	Ground wire connected to the ground of the system.
Red	Powers the motor (typically +5V is used).
Orange	PWM signal is given in through this wire to drive the motor

2.4.2 Specifications

- Operating Voltage: +5V (typical).
- Torque: 2.5kg/cm.
- Operating speed: 0.1s/60°.
- Gear Type: Plastic.
- Rotation : 0°-180°.
- Weight of motor : 9gm.
- Package includes gear horns and screws. [6]

2.4.3 Mechanism

A servo motor is a closed loop mechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analog or digital) representing the position commanded for the output shaft. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.[7]

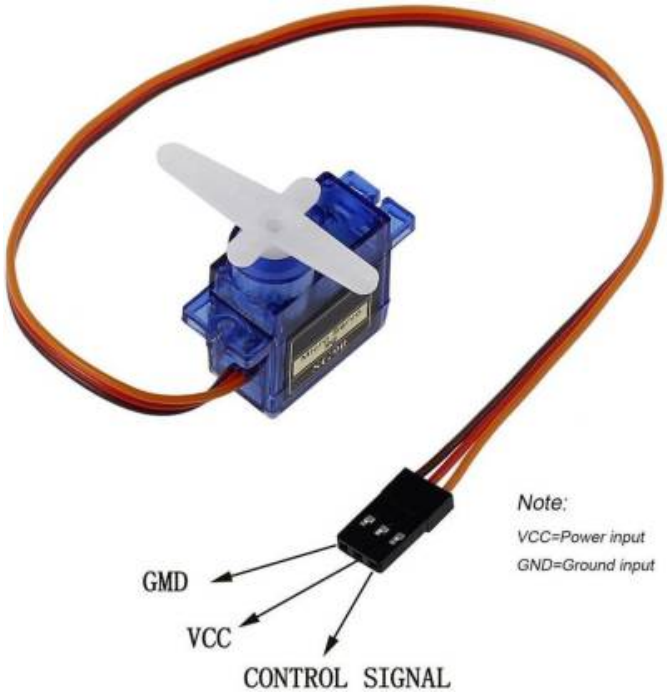

Figure 2.4: The 9g servo

2.5 Stepper Motor

A Stepper Motor or step motor is a brushless, synchronous motor, which divides a full rotation into a number of steps. Unlike a brushless DC motor, which rotates continuously when a fixed DC

voltage is applied to it, a step motor rotates in discrete step angles. Model of the stepper motor used in our project: 28BYJ-48

2.5.1 Wire configuration

A stepper motor has a total of 4 coils. One end of all the coils is connect to +5V (the red wire) and the other end of each coil is pulled out as colored wire (orange, pink, yellow and blue respectively).[8]

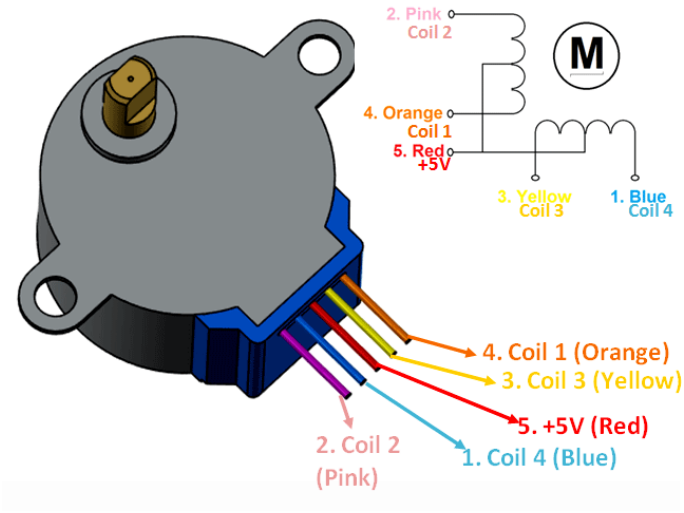

Figure 2.5: 28BYJ-48 Stepper

2.5.2 Specifications

- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Pull in torque: 300 gf.cm
- Insulated Power: 600VAC/1mA/1s
- Coil: Unipolar 5 lead coil
- Steps per revolution: 2048

2.5.3 Mechanism

Stepper motors can turn an exact amount of degrees (or steps) as desired. This gives total control over the motor, allowing to move it to an exact location and hold that position. It does so by powering the coils inside the motor for very short periods of time.
To move a stepper motor, the controller needs to know the rotation speed, number and direction of steps. N.B. The disadvantage is that the motor draws power to hold its position.

2.6 Breadboard

A breadboard is a solderless construction base used for developing an electronic circuit and wiring for projects with microcontroller boards like Arduino.

2.6.1 Construction

A breadboard consists of two areas called strips: the bus and terminal strip.
• Bus strips are mainly used for power supply connections
• Terminal strips are mainly used for electrical components. Each strip consists of 5 pinholes, indicating that you only can connect up to 5 components in one particular section
Note how the holes colored in yellow are connected together. These sets of connecting holes can be called a node, where it's possible to interconnect the node from bus strips to terminal strips with jumper wires.

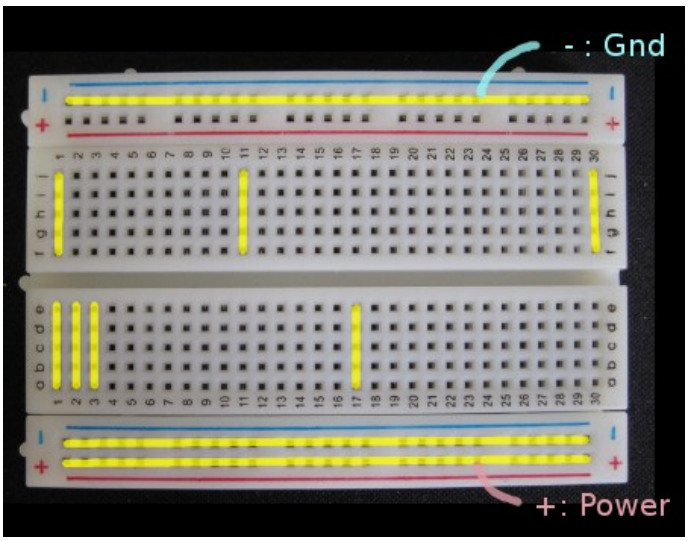

Figure 2.6: Mini breadboard

2.7 Development environment

The Arduino IDE (Integrated Development Environment) makes it easy to write code and upload it to an Arduino board.

2.7.1 Programming Language Used

The Arduino compiler/IDE accepts C and C++ as-is. In fact many of the libraries are written in C++. Much of the underlying system is not object oriented, but it could be. Thus, "The Arduino language" is C++ or C.

2.7.2 Getting started

a. Open Arduino app.
b. Go to file menu and select new page.
c. Write program.
d. Compile.
e. Upload the binary to Arduino.

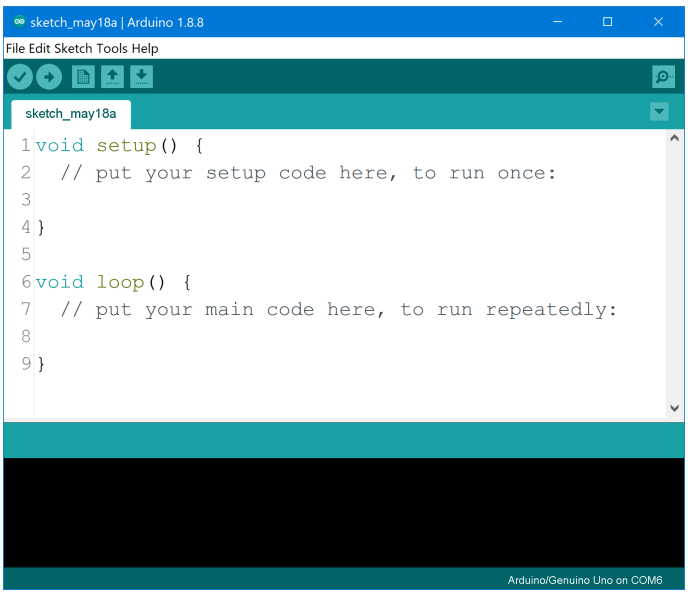

Figure 2.7: IDE startup interface

Chapter 3

Hardware Assembly

3.1 Circuit Design

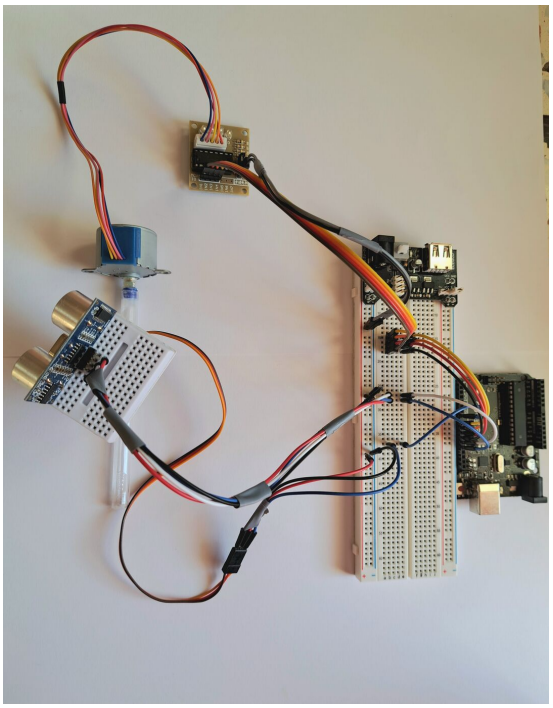

Figure 3.1: Circuit assembly

3.2 Connections

Component	Arduino pin number
Ultrasonic trig pin	12
Ultrasonic echo pin	13
Servo control pin	7
Stepper coil 1	8
Stepper coil 2	9
Stepper coil 3	10
Stepper coil 4	11

Chapter 4

3D imaging

4.1 Introduction

To get a basic 3D image using a distance sensor, we need to rotate and tilt it in precise angles and measure the distance from each position. The data can be used to estimate the dimensions and depth of a surface.

4.2 Collecting data

In our assembly, we will assume that $x'Ox$ is along the shaft of the stepper motor, $z'Oz$ is along the axis supporting the servo. The plot origin is taken at the intersection between the previous axis-es. To get 3D coordinates:
- The stepper motor will tilt the assembly around the $x'Ox$ axis in the direction of negative θ in increments of 2.1 degrees, until reaching 37.8 degrees.
- The servo will sweep from 45 to 135 degrees back and forth in increments of 5 degrees.
- After each servo motor movement, the ultasonic sensor will activate and measure the distance. The combination of distance, servo angle and stepper tilt are enough to calculate a data point.
- The data point will be sent over the USB ACM serial connection to the computer for processing (format: Distance,Angle,Elevation).
Where Distance is the distance from the sensor to the object, "angle" is equivalent to ϕ in spherical coordinates and elevation is the angle of the assembly relative to the $z'Oz$ axis.

4.3 Processing data

While the data collected earlier is quite similar to spherical coordinates, it doesn't fully map to them. See 4.1 below:

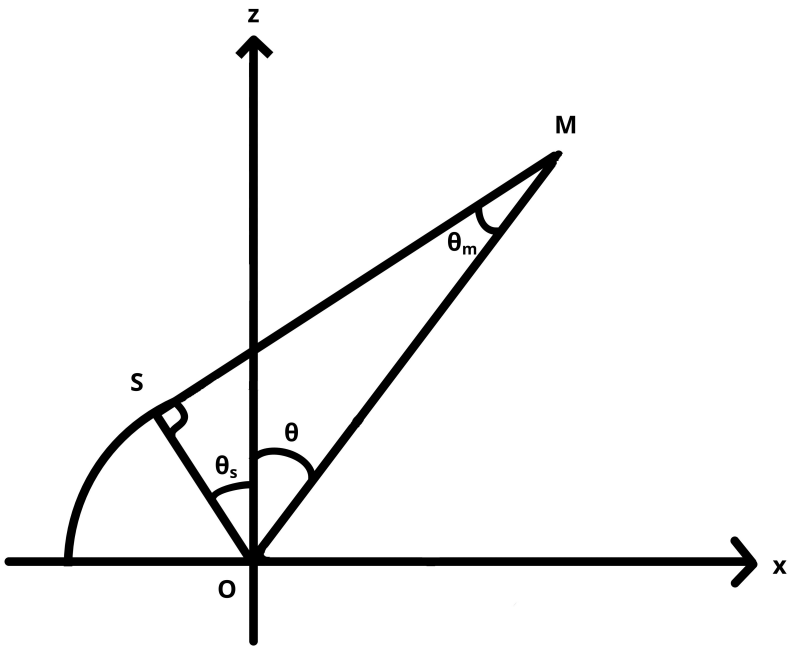

Figure 4.1: The measurements at $\phi = 0$

The tilt angle θ_s becomes almost complementary to the spherical coordinate θ when the object M is relatively far, because θ_m becomes pretty small and the angle OSM is 90°. Without correction, this will toggle the sines and cosines in the Cartesian coordinates calculation.
As for ρ, it is equal to OM, however the sensor is measuring SM.
θ being the angle between $x'Ox$ and OM' (M' the projection of M on xOy) is not affected by the rotation of OS, because it is rotating around the $x'Ox$ axis.
In conclusion, we need to get OM and θ, knowing OS, SM and θ_s.

First, get ρ using Pythagorean theorem in OSM:

$$\rho = \sqrt{OS^2 + SM^2} \tag{4.1}$$

Next, use ρ to get θ_m using basic trigonometry in the triangle OSM:

$$\theta_m = \sin^{-1}\left(\frac{OS}{\rho}\right) \tag{4.2}$$

Finally:

$$\theta = 180^\circ - 90^\circ - \theta_m - \theta_s \tag{4.3}$$

The computer will compute the corrected spherical coordinates then the Cartesian coordinates and save them in a file. The points will be 3D rendered using GNUplot.

4.4 Scan Samples

4.4.1 Book sound shadow

An open book was placed on a table in front of a wall. The assembly was placed about 0.5 meters behind the book. Figure 4.2 shows that the book did cast a ”shadow” on the wall behind it, which

is consistent with the rectangular shape of the book. The slight curvature of the wall image is possibly due to imperfections in measured angles and slight slipping on the stepper shaft.

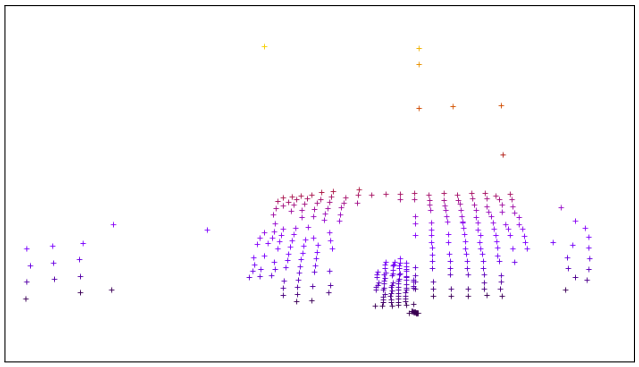

Figure 4.2: Book shadow - Front

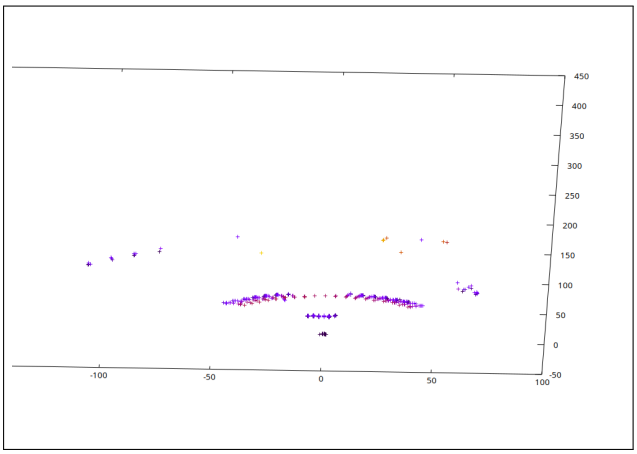

Figure 4.3: Book shadow - Top

4.4.2 Cordless phone

Another sample is scanning a cordless phone:

Figure 4.4: Cordless phone to scan

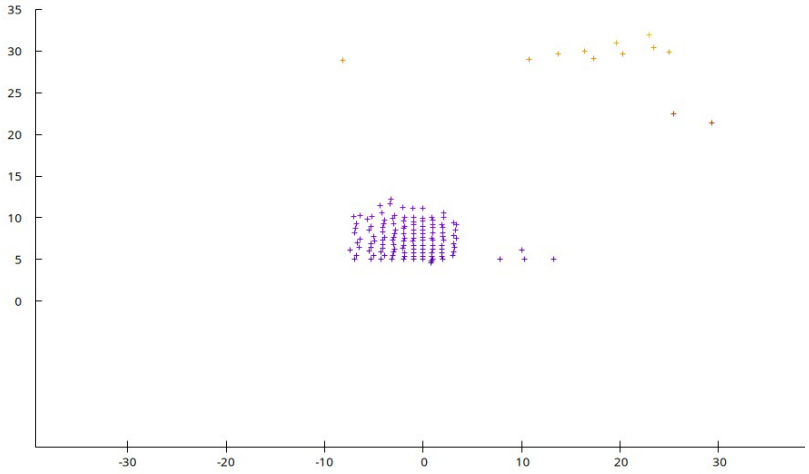

Figure 4.5: Cordless phone scan results

Taking the relevant section of the data points (removing the wall points), we can see a rectangular surface of dimensions 12cm*8cm which roughly matches the actual dimensions. Three data points did hit the handset upper part, but other data points missed it and continued into the wall. Interestingly, when the scanned surface is making more than about 40° with the ultrasonic sensor, the distance reading becomes far larger than expected. A possible explanation is that sound is bouncing off the surface and not directly back to the sensor, instead it is taking a longer trip to return.

Conclusion

In conclusion, it is possible to use a distance sensor to obtain a basic 3D image of an object. However, the image is of very low resolution due to very small number of data points and the hardware limitations (relying on time only may be fooled by the reflection path). This design can be improved by increasing the sweeping speed and number of data points.

Bibliography

[1] Sonar - Wikipedia

[2] UNO R3 — Arduino Documentation

[3] HC-SR04 Ultrasonic Sensor Guide with Arduino Interfacing

[4] HC-SR04-Ultrasonic.pdf User Guide

[5] Micro bit Lesson — Using the Ultrasonic Module

[6] Servo Motor SG-90 Basics, Pinout, Wire Description, Datasheet, and Working

[7] Servo Motor - Electronics Infra

[8] 28BYJ-48 Stepper Motor Pinout Wiring, Specifications, Uses Guide & Datasheet

Appendix

Arduino driver code

```
#include <Stepper.h>
#include <Servo.h>

const int trigPin = 12;
const int echoPin = 13;
const int stepsPerRevolution = 2048;
bool isReversed = false;
float Distance = 0;
int Angle = 0;
float Elevation = 0;

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);
Servo myServo;

void setup()
{
    Serial.begin(9600);
    myServo.attach(7);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    myStepper.setSpeed(10);
}

void loop()
{
    delay(5000);
    Serial.print("start \n \n");
    for (int i = 0; i < 18; i++)
    {
        Elevation = (360 * 12.0 / 2048) * i;
        servoSweep();
        getDistance();
        myStepper.step(-12);
        delay(500);
    }
    Serial.print("\nEnd \n \n \n");
    myStepper.step(18 * 12);
```

```
    Elevation = 0;
}

float getDistance()
{
    // Generate a pulse by switching from low to high for 10 microseconds
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // The ultrasonic sensor echo pin remains high until the sound echo is received
    // Get the duration of the high pulse on the echo pin
    unsigned long duration = pulseIn(echoPin, HIGH);
    // Speed of sound considered to be 340m/s (divided by 2 because it is a round trip)
    float distance = duration * 0.034 / 2;

    // Discard readings larger than 4 meters as the sensor is not rated for higher values
    if (distance <= 400)
        return distance;
    else
        return -1;
}

void printCoordinates()
{
    if (Distance > 0)
    {
        Serial.print(Distance);
        Serial.print(",");
        Serial.print(Angle);
        Serial.print(",");
        Serial.println(Elevation);
    }
}

void servoSweep()
{
    // The servo sweeps back and forth between 0 and 90 degrees
    // isReversed determines the rotation direction
    if (isReversed == false)
    {
        for (Angle = 0; Angle <= 90; Angle += 5)
        {
            myServo.write(Angle);
            // Wait the servo to get in position
            delay(250);
            Distance = getDistance();
            printCoordinates();
```

```
        }
        isReversed = true;
    }
    else
    {
        for (Angle = 90; Angle >= 0; Angle -= 5)
        {
            myServo.write(Angle);
            delay(250);
            Distance = getDistance();
            printCoordinates();
        }
        isReversed = false;
    }
}
```

Data processing and plotting code

Expects a Linux based system.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>

#define POINTS_PER_SWEEP 19
#define STEPPER_TOTAL_STEPS 18
#define TOTAL_DATAPOINTS (POINTS_PER_SWEEP * STEPPER_TOTAL_STEPS)
#define SENSOR_HEIGHT_CM 5

int main()
{
    FILE *serial_input = NULL, *output = NULL, *locate_stream = NULL;
    char tmp[256];
    double x, y, z;
    double ro, teta, phi;
    int i, status;

    puts("Waiting for input ... ");
    while (serial_input == NULL)
    {
        // Detect the USB serial connection by scanning /dev
        locate_stream = popen("echo /dev/`ls /dev | grep ttyACM`", "r");
        fscanf(locate_stream, "%255s", tmp);

        // If the output is not /dev/
        if (strlen(tmp) > 5)
```

```
        serial_input = fopen(tmp, "r");

        pclose(locate_stream);
        sleep(1);
    }
    printf("Input stream found, reading from %s\n", tmp);

    do
    {
        fscanf(serial_input, "%255s", tmp);
        sleep(1);
    } while (strcmp(tmp, "start") == 0);

    puts("Received start signal, reading and processing data...");
    // Open the output file
    output = fopen("plot_data.txt", "w");

    i = 0;
    while (i < TOTAL_DATAPOINTS)
    {
        fscanf(serial_input, "%255s", tmp);
        status = sscanf(tmp, "%lf,%lf,%lf", &ro, &phi, &teta);
        if (strcmp("End", tmp) == 0)
        break;
        // Skip empty inputs
        if (status == 3)
        {
            teta *= M_PI / 180;
            phi += 45;
            phi *= M_PI / 180;
            // Correcting ro
            ro = sqrt(pow(SENSOR_HEIGHT_CM, 2) + pow(ro, 2));

            double teta_c = asin(SENSOR_HEIGHT_CM / ro);

            // Correcting teta
            teta = M_PI / 2 - teta_c - teta;

            x = ro * sin(teta) * cos(phi);
            y = ro * sin(teta) * sin(phi);
            z = ro * cos(teta);
            printf("P%d (%g,%g,%g)\n", i, x, y, z);
            fprintf(output, "%lf %lf %lf\n", x, y, z);
            ++i;
        }
        // Prevent thrashing the computer performance if the serial line is not sending data,
            wait 0.1s between reads
        else
        usleep(100000);
    }
```

```
    fclose(output);
    // Call the plotter
    system("echo \"splot '$PWD/plot_data.txt' with points palette\" | gnuplot --persist");
    return 0;
}
```