IIIT- Hyderabad

Electronic Workshop-II

SHAPE DETECTION OF THE OBJECTS

Names :

Sachin Chandani & Sushman Kvs.

Roll Numbers

20161201 & 20161143.

Course Instructors

Prof. Zia Abbas.

Prof. Syed Azeemuddin.

Mentor

Prof. Lavanya Ramapantulu.

INDEX

Introduction

Object recognition

Technology in the field of computer vision and electronics for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different view points, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

Object detection is a computer technology related to computer vision, electronics and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection. Object detection has applications in many areas of computer vision, including image retracing and motion detection.

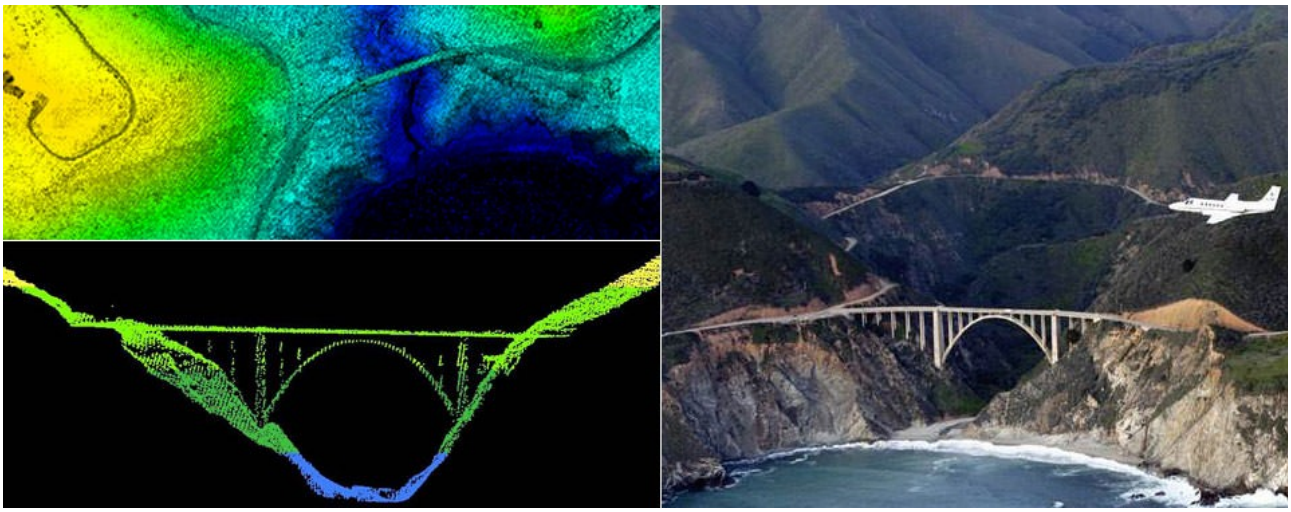
Our objective of this project is to detect the shape of simple projects using simple electronics equipments like IR sensors and micro-controllers and process their 3 Dimensional image image on a screen by determining the coordinates of differents parts of the object.

MOTIVATION – LIDAR(Light Detection And Ranging)

LiDAR or **Li**ght **D**etection **a**nd **R**anging is an active remote sensing system that can be used to measure vegetation height across wide areas.

Why LiDAR

Scientists often need to characterize vegetation over large regions to answer research questions at the ecosystem or regional scale. Therefore, we need tools we need tools that can estimate key characteristics over large areas because we don’t have the resources to measure each and every tree or shrub.



Remote sensing means that we aren’t actually physically measuring things with our hands. We are using sensors which capture information about a landscape and record things that we can use to estimate conditions and characteristics. To measure vegetation or other data across large areas, we need remote

sensing methods that can take many measurements quickly, using automated sensors.

LiDAR, or light detection ranging (sometimes also referred to as active laser scanning) is one remote sensing method that can be used to map structure including vegetation height, density and other characteristics across a region. LiDAR directly measures the height and density of vegetation on the ground making it an ideal tool for scientists studying vegetation over large areas.

How LiDAR Works

LiDAR is an **active remote sensing** system. An active system means that the system itself generates energy - in this case, light - to measure things on the ground. In a LiDAR system, light is emitted from a rapidly firing laser. You can imagine light quickly strobing from a laser light source. This light travels to the ground and reflects off of things like buildings and tree branches. The reflected light energy then returns to the LiDAR sensor where it is recorded.

A LiDAR system measures the time it takes for emitted light to travel to the ground and back. That time is used to calculate distance traveled. Distance traveled is then converted to elevation. These measurements are made using the key components of a lidar system including a GPS that identifies the X,Y,Z location of the light energy and an Internal Measurement Unit (IMU) that provides the orientation of the plane in the sky.

Principle – Distance sensing using IR sensor and microcontroller

Theory

At the heart of this sensor there are two IR LEDs. One emits light, the other receives it.When an object is close to the sensor, it will inevitably reflect some of the IR light. This is detected by the Arduino, and translated into a distance measurement. The Human eye does not pick up infrared light; this sensor will look invisible, even though it can be extremely bright in the IR spectrum.There are many things that can go wrong with this approach; IR light can be erroneously picked up from another source, or the objects is not reflective enough. This picture shows a face of the DodecaLEDron, an alternative game controller which is based on the same principle.

Circuit

This circuit has three LEDs.The bottom ones are IR emitters ; the top one is an IR photodiode receivers. It is important to use *photodiodes*, and not *phototransistors*. Both types are IR receiver which change their resistance according to how much light they sense. However, while the latter works like a digital switch: it either sense IR or it doesn’t. For this project we need a photodiode because we actually need to measure the quantity of light received.

To measure the amount of light sensed from the IR receiver, we measure the drop of voltage on its resistor. When the photodiode receives light, it allows current to flow from the 5V to the GND pin; little to no current then flows back to A0.On the other hand, when the diode is in darkness and doesn't let current flow, A0 reads a high voltage. For this project you have to use an *analog input* (A0-A5).

Calliberartion

The moist naive way you can read data from the circuit is the following:

```
void setup () {
 pinMode(A0, INPUT);
}

void loop () {
 int distance = analogRead(A0)
 Serial.println(distance);
}
```

Most of the tutorials we found which explains how to do IR distance sensors stop here.As software aspects, We believe we can do much, much more on the software side.First of all, the distance readings of this sensor are likely to be affected by the IR background noise in your room.Almost every artificial light source is visible in the IR spectrum, causing interferences with our sensor.

sensor when (1)no object is nearby and (2) an object is very close. This provides a baseline calibration which depends on the light in the room.

A more sophisticated approach to solve this problem is to calibrate the light background in your environment.

```
int duration = 1000;

int calibrationZero;
int calibrationOne;

void setup () {
 pinMode(sensorPin, INPUT);

 // Far
 Serial.print("Far calibration...");
 delay(duration);

 int sensorCumulativeValue = 0;
 int reads = 0;
 unsigned long timeStart = millis();
 do {
 delay(20);
 sensorCumulativeValue += analogRead(sensorPin);
 reads ++;
 } while (millis() <= timeStart + duration);
 calibrationZero = sensorCumulativeValue / reads;

 Serial.print(calibrationZero);
 Serial.println();

 // Close
 Serial.print("Close calibration...");
 delay(duration);

 sensorCumulativeValue = 0;
 reads = 0;
 timeStart = millis();
 do {
 delay(20);
 sensorCumulativeValue += analogRead(sensorPin);
 reads ++;
 } while (millis() <= timeStart + duration);
 calibrationOne = sensorCumulativeValue / reads;

 Serial.print(calibrationOne);
 Serial.println();
}
```

This new setup function is divided in two seconds, to calibrate the sensor for far and nearby objects, respectively. The variable duration determines how long each calibration phase lasts. This is important because single readings are prone to error, but averaging them over a longer time interval provides more reliable data. The result of the calibration is stored in calibrationZero and calibrationOne, which will be used later to correct our readings.

The reading :

Reading from the sensor works in a similar fashion. We repeat a certain numbers of readings over a long time interval, and average them out to make sure about their reliability.

```
1  int getDistanceReading (int readings = 5) {
2   int sensorCumulativeValue = 0;
3   for (int i = 0; i < readings; i ++) {
4   delay(20);
5   sensorCumulativeValue += analogRead(sensorPin);
6   }
7
8   int distance = map
9         (
10          sensorCumulativeValue / readings,
11          sensorCalibrationZero, sensorCalibrationOne,
12          0, 255
13        );
14  return constrain(distance, 0, 255);
15 }
```

Lines 8-13 use the function map to rescale the readings between 0 and 255. These two values corresponds to the amount of IR light received during the two calibration steps.

Finally, to use this code :

```
2  int distance = getDistanceReading ();
3  Serial.println(distance);
4 }
```

STRUCTURE

As defined in previous section, our system works on the principle of distance sensing of different parts of object and plotting them as a dot graph on a 3Dimensional graph and generating the 3Dimensional shape of that object.

Circuit Structure

The circuit structure conatins a 6 x 4 board in dimension and 1 emitter/detector pair in each cell.

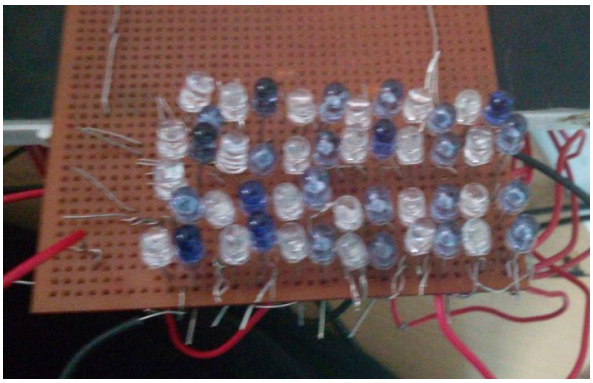
Front view of the circuit(6x4 board):

O	0	O	0	O	0	O	0	O	0	O	0
O	0	O	0	O	0	O	0	O	0	O	0
O	0	O	0	O	0	O	0	O	0	O	0
O	0	O	0	O	0	O	0	O	0	O	0

0 = denotes senor

O= emitter

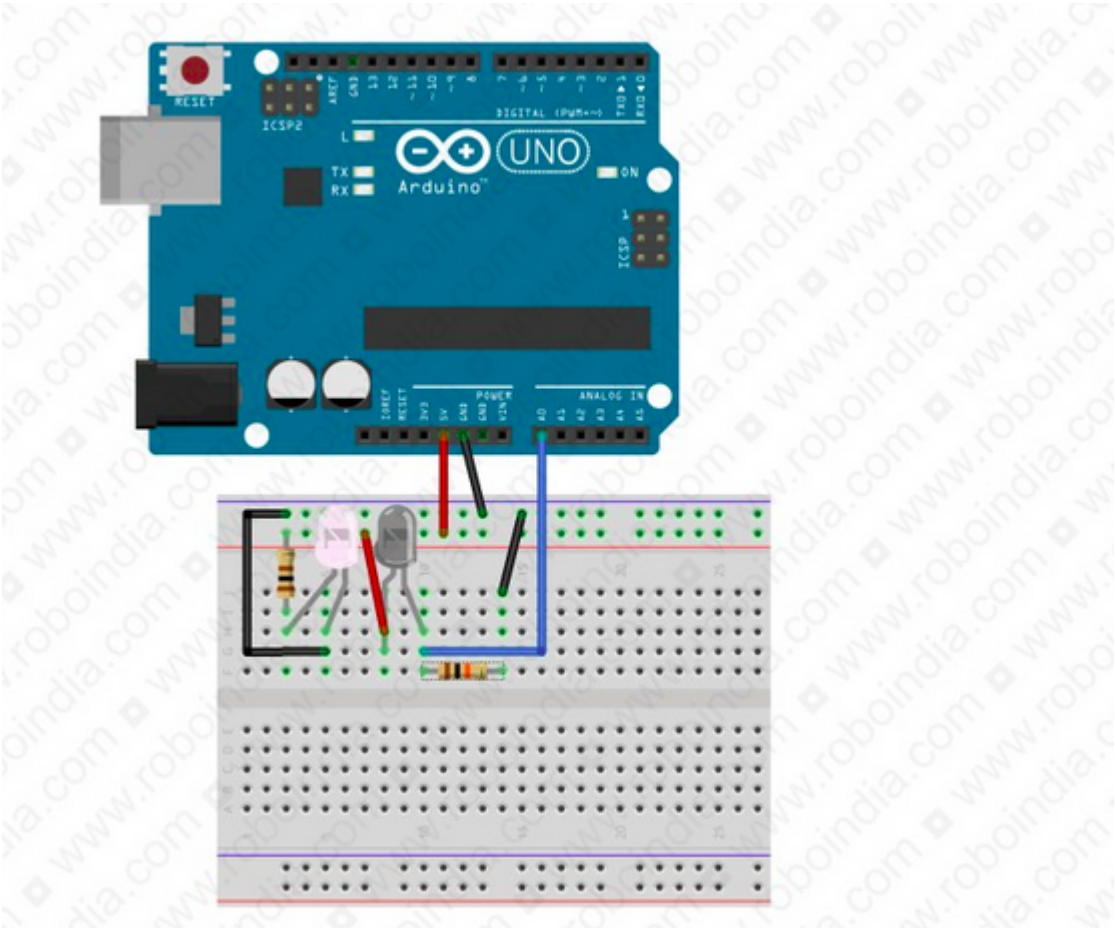
Actual Picture :



Blue(dark) diode = sensor.
Transparentdiode=emmiter.

Working Circuit for Each Emitter-Detector Circuit

For each IR Emitter-Detector Circuit, we need one 220 Ohm and one 10k Ohm resistor for emiiter and detector.



For 6 x 4 =24 diodes to operate simultaneously, we created a 5V bus and Gnd bus to provide volatge to simultaneouly all the diode pairs.

At a single period of time, max of 6 diodes can be operated so a switch mechanism will be used to operate one layer of diode(6 diodes at a time) at a time and in 4 cycles, all the 24 coordinates will be recorded.

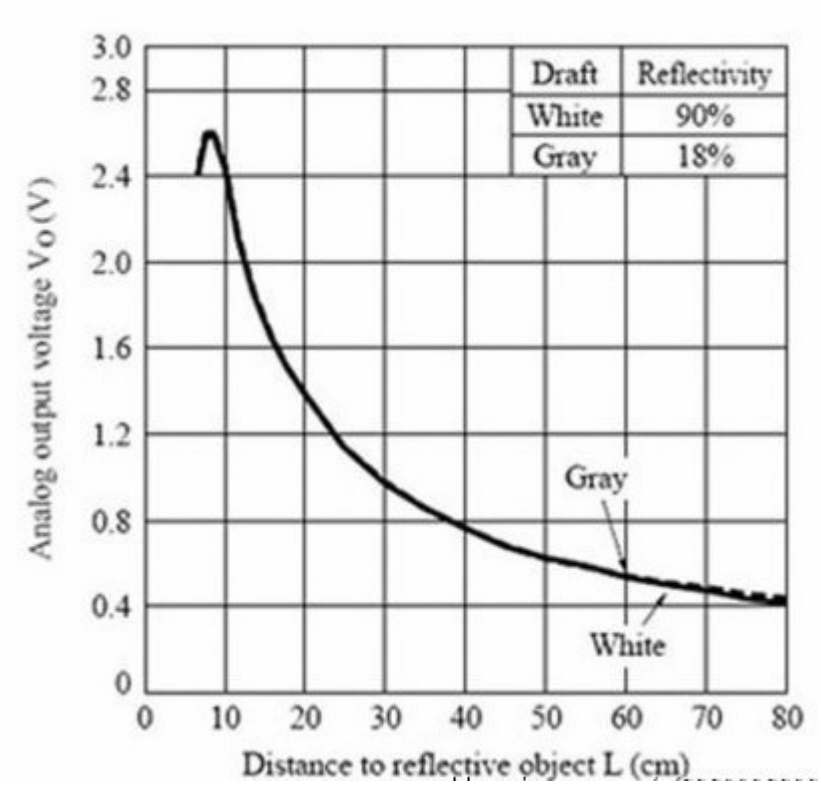
WORKING

Calibration of All the LEDs

All 24 LEDs need to calibrated as there will be fluctuation in the number obtained from each LED pair.

These LEDs are calibrated by measuring the changed in observed number per unit centimeter and measuring the linear range of each LED.

The variations in the voltage of photdiode is shown :



*****The voltage is inversly propotional to distance of the object .

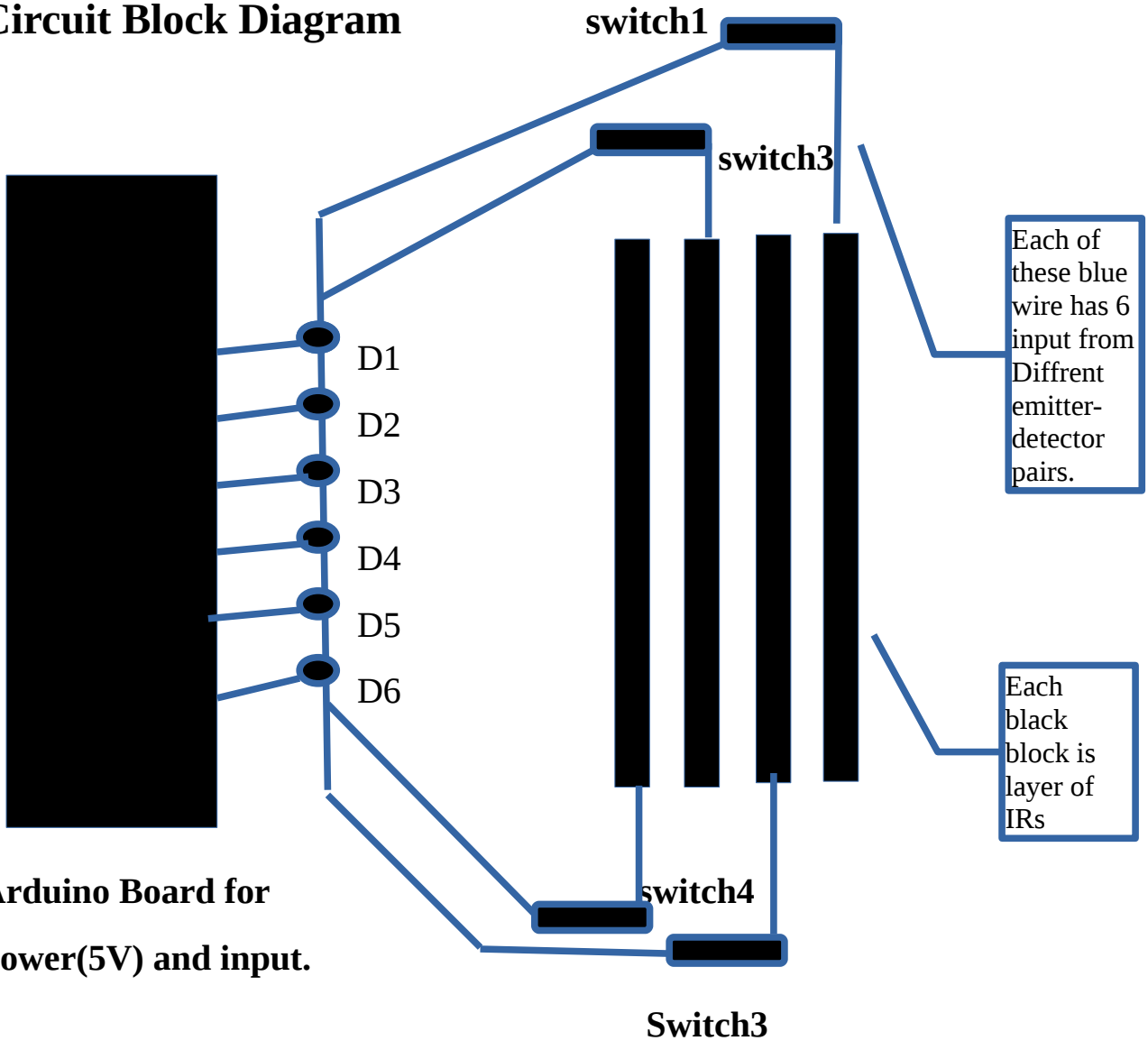
Working

After caliberating all the pairs the working of the circuit is explained by the following steps :

1. Connect 5V and Gnd bus to the 5V and GND of arduino board.

*P.T.O

Circuit Block Diagram



2. Switch1 = ON, Switch2 = OFF, Switch3 = OFF, Switch4 = OFF and load the code and fill the first row of y=4 of the data table.

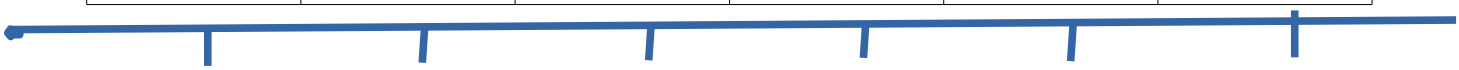
3. Switch1 = OFF, Switch2 = ON, Switch3 = OFF, Switch4 = OFF and load the code and fill the first row of y=3 of the data table

---

The data will be recorded as a 6x4 graph like : (Xi,Yi,Zi)

(1,4,Z1)	(2,4,Z2)	(3,4,Z3)	(4,4,Z4)	(5,4,Z5)	(6,4,Z6)
(1,3,Z7)	(2,3,Z8)	(3,3,Z9)	(4,3,Z10)	(5,3,Z11)	(6,3,Z12)

(1,2,Z13)	(2,2,Z14)	(3,2,Z15)	(4,2,Z16)	(5,2,Z17)	(6,2,Z18)
(1,1,Z19)	(2,1,Z20)	(3,1,Z21)	(4,1,Z22)	(5,1,Z23)	(6,1,Z24)



X= 1 **X =2** **X=3** **X=4** **X=5** **X=6**

*Where each Zi is the distance of part of object calculated by that particular sensor.

Xi, Yi will be a fixed value for each LED.

The values of Zi will stored as -Zi as plotter will axis is coming out of the screen.

This data will be stored in a [7][5] array data structure(1 indexing)

---

4. Switch1 = OFF, Switch2 = OFF, Switch3 = ON, Switch4 = OFF and load the code and fill the first row of y=2 of the data table.

5. Switch1 = OFF, Switch2 = OFF, Switch3 = OFF, Switch4 = ON and load the code and fill the first row of y=1 of the data table.

6. Send this table data to a 3D plotter to form a dottted 3 Dimensional shape of the object.

RESULTS

Image Forming

1. When this table of data is passed to 3 Dimensional dot plotter, an outline of shape of object will be obtained.

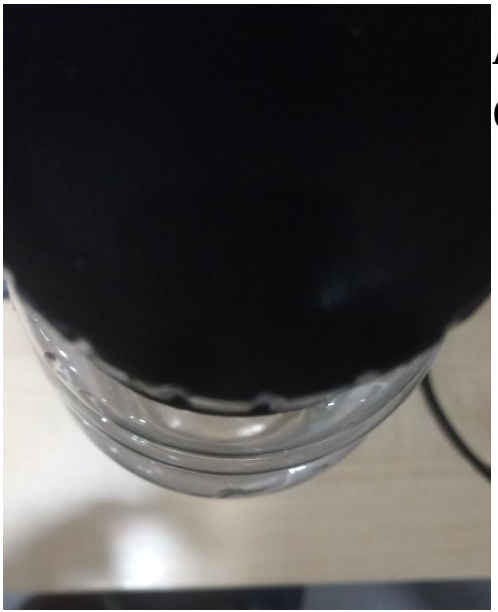
We ran our system on a plastic bottle which has cylindrical in nature

The data table was obtained like this :

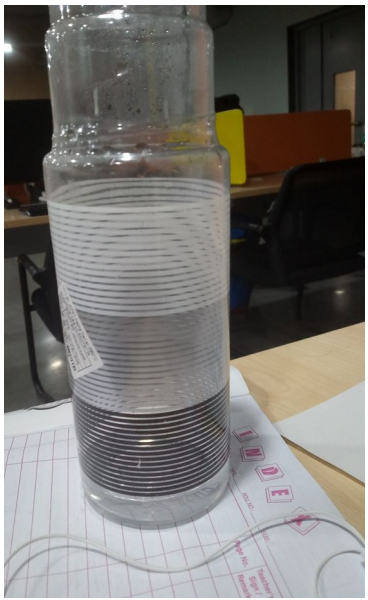
(1,4,-4)	(2,4,-3)	(3,4,-2)	(4,4,-2)	(5,4,-3)	(6,4,-4)
(1,3,-4)	(2,3,-3)	(3,3,-2)	(4,3,-2)	(5,3,-3)	(6,3,-4)
(1,2,-4)	(2,2,-3)	(3,2,-2)	(4,2,-2)	(5,2,-3)	(6,2,-4)
(1,1,-4)	(2,1,-3)	(3,1,-2)	(4,1,-2)	(5,1,-3)	(6,1,-4)

This data was pushed into CPM 3D plotter :

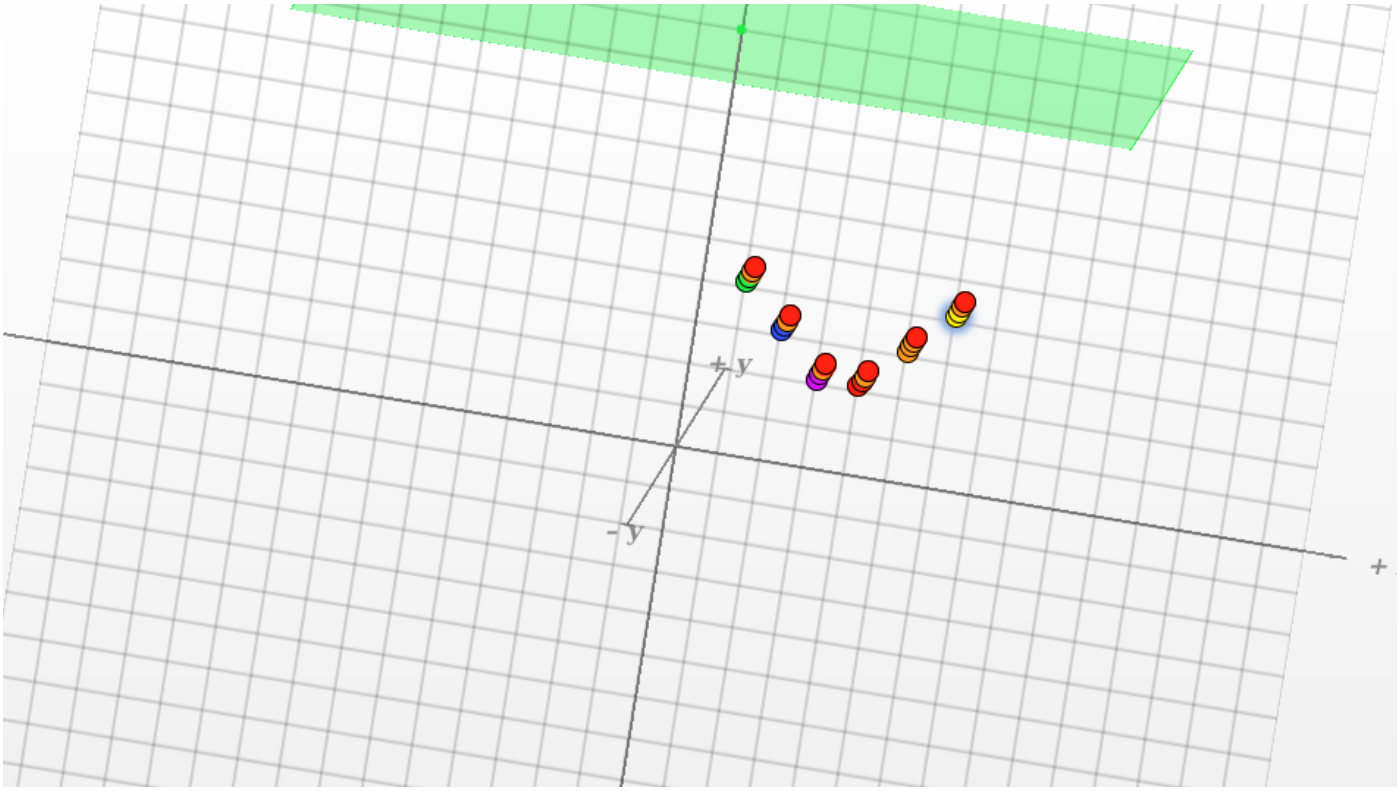
link to output : https://technology.cpm.org/general/3dgraph/?graph3ddata=cJxt2x1Yv48KxJcx1YwkiLxYmx1YwrUMx9wx1YwkiKyoGx1Yv48Lxmux1Yv-AMxBEx1YwcKHxQOx1YwrUIx1Yx1YwrUJyg4x1YwcKKywex1Yv-Afw7kw7kxmudD7E



Actual Object

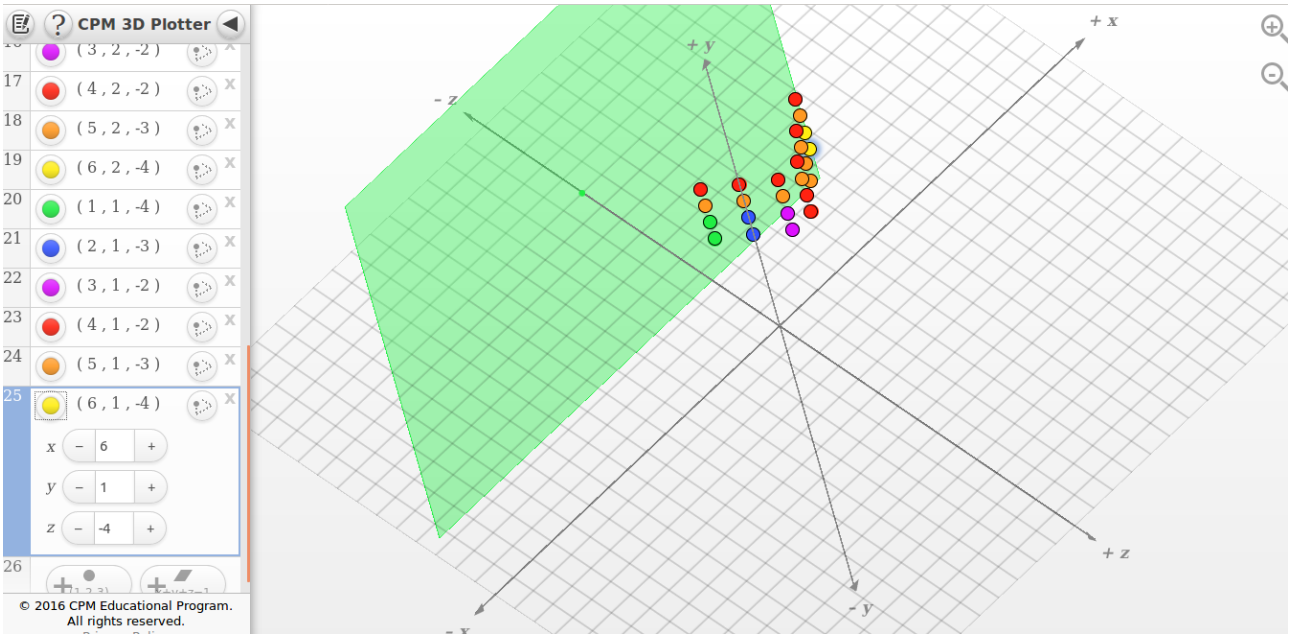


The Output of the shape detector



Output 1

The Output shows the shape- curvature of the bottle till 6 cm width and 4 cm height.



Output 2

POSSIBLE MODIFICATIONS AND COST

Cost : (40RSxnum_sensors)+(1x500(arduino))+Extras

Current Cost : 40x24 + 500 + 100 = 1560.

To Cover up large objects and more accuretly

We have used a 6x4 board i.e. 24 diodes which cannot cover up objects having diameter(max internal distance > 6), hence by increasing the number of diodes and placing them close will increase its efficiency.

Improving Cost and Accuracy by using a Motor

The cost and accuracy can be drastically improved by using a motor to operate a single layer of diodes from y=n , y= n-1 ..... till y=1.

Hence more diodes can be placed in single layer and can be moved from top to down to record the data for each row of table.

Cost : (40RS x num_sensors) + 500(arduino1 for data recording) + 500(Arduino2 for motor movement) + 300(extras) = 1300 + 40xn

*where n = number of sensors

Hence by introducing a motor, more accurate results can be obtained.

Thank You :)