SAFETY ASSISTANCE SURVEILLANCE BOT

PROJECT REPORT

Submitted for the course: Robotics and Automation
(ECE2008)

By

NAME	REG NO.	SLOT
KARVEANDHAN P	17BEC0003	E1
S.ARUN KARTHIK	17BEC0114	E1
MAKADIA SHIV SANJAYBHAI	17BEC0227	E1

NAME OF FACULTY: PROF. BUDHADITYA BHATTACHARYYA

(SCHOOL OF ELECTRONICS ENGINEERING**)**

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

INTRODUCTION

Security aspects are the main focus of the project. There is a need for high surveillance especially in sensitive areas, military borders, government offices, public places, security check points and even at homes. A robot can help in outdoor surveillance by monitoring important places and also it can be made to reach suspicious places for close observations. In risky and dangerous situations such as rescue operations, a robot could be a great help to the rescue team and can greatly improve the safety of the rescue team as well as their working efficiency. A robot capable of moving in all types of terrain mounted with the camera and sensors can go to the affected area and can provide all the live information to the rescue team, as per the situation the rescue team can plan its activity.

METHODOLOGY

- The bot is controlled by using a microcontroller – Arduino.
- The bot will be given instructions using an android app by using an android device and will send it to a Bluetooth module.
- Bluetooth module in turn sends these signal to Arduino using serial communication. Arduino will then drive the Bot using L293D motor driver.
- The bot will be giving the input of its surroundings by an ultrasonic sensor acting as a 3D mapper and the camera of the mobile phone mounted on it to give Video Feed.
- This will give a proper surveillance of the surroundings.
- The bot now can be tracked using its GPS location which can be found using the phone mounted on the device.

In this project we are going to Control the Robot Car through the G sensor of our mobile phone and you will be able to move the Robot just by tilting the Phone. We will also use Arduino and Android Apps and Processing software developed for the project. This app is used to create the interface in the Smart Phone for controlling the Robot. We will add the joystick in the interface so that Robot can also be controlled by Joystick as well as by tilting the phone.

G-Sensor or Gravity sensor is basically Accelerometer in smartphones which is used to control the screen orientation of the phone. Accelerometer senses the X, Y,Z directions of the Gravitational force and rotate the Screen according to alignment of the Phone. Nowadays, more sensitive and accurate Gyroscope sensor is used in mobiles for deciding the orientation of the Screen.

In our Project, Robot car will move, according to the direction in which phone is being tilted, like when we tilt the phone forward, then car will move forward and we tilt it down then car will move backward. This is same like when we play some car games in Mobile, they also use G sensor to move the car accordingly.

TECHNICAL BLOCKS OF THE PROJECT:

- Mobility Block:

ATMEGA328P Microcontroller then generate signals to drive the motors in clock wise and anticlockwise direction using L293D motor driver IC. Using a mobile application the user sends the data to the arduino using Bluetooth communication.

- Video Surveillance Block:

Camera captures the video and sends the data in form of images using wifi communication which can be viewed on a display device.

- Range Finder Block:

The ultrasonic sensors captures the distance of the obstacles and sends it to a display device(laptop) using bluetooth communication.

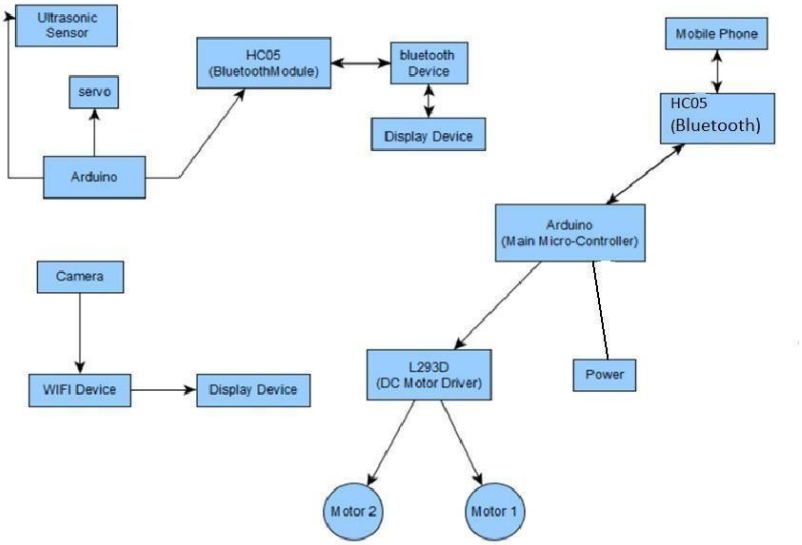
COMPONENTS REQUIRED

Hardware: L293D, Arduino 1DE, Bluetooth module; camera; IR sensors; ultrasonic sensors; DC motors; servomotors.

Software Development:

1. The microprocessor "Arduino Uno "is coded using Arduino IDE and the language used is Embedded C.
2. Android Studio is used to develop an app that controls the L293Dprogrammed in JAVA.
3. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions
4. The Data feeder by the camera will be uploaded on the serverby using Blank free server
5. Processing is used to map the data obtained from the ultrasonic sensor

BLOCK DIAGRAM



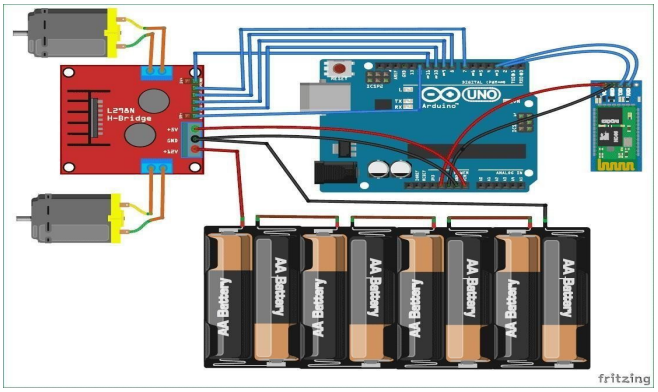
WORKING

Controlling Of Bot (Mobility Block)

First of all, we will interface the L293D motor controller with the Arduino. Then connect the IN1, IN2, IN3 and IN4 to the Arduino pins 10, 9, 8 and 7 respectively. These pins will rotate the motors in both directions (clockwise and anti-clockwise).

To power the motor, connect the positive and negative of the battery to the 12V and the ground of the motor controller. Then connect the 5V and the ground from the motor controller to the Arduino Vin and the ground.

Then we will connect the Bluetooth module HC-05 with the arduino. Connect the VCC and the ground of the Bluetooth module to the 5V and the ground of the Arduino. Then connect the TX pin of Bluetooth Module to the pin 2 of Arduino and the RX pin to the pin 3 of Arduino.



Circuit diagram of the mobility block

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC.

It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. In a single L293D chip there are two h- bridge circuit inside the IC which can rotate two dc motor independently.

L293D Logic Table

Let's consider a Motor connected on left side output pins (pin 3,6). For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

- Pin 2 = Logic 1 and Pin 7 = Logic 0 | Clockwise Direction
- Pin 2 = Logic 0 and Pin 7 = Logic 1 | Anticlockwise Direction
- Pin 2 = Logic 0 and Pin 7 = Logic 0 | Idle [No rotation] [Hi-Impedance state]
- Pin 2 = Logic 1 and Pin 7 = Logic 1 | Idle [Norotation]

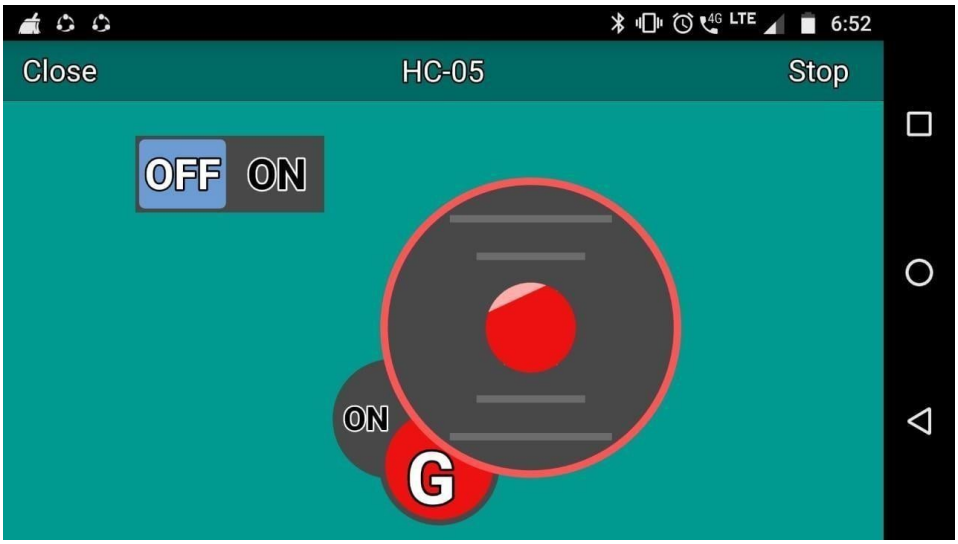
So to move in a particular direction:

Direction	Left motor	Right motor
Forward	Clockwise	Clockwise
Backward	Anti-clockwise	Anti-clockwise
Left	Clockwise	Anti-clockwise
Right	Anti-clockwise	Clockwise

The information received by the joystick is transmitted to the arduino via Bluetooth communication using HC05 bluetooth module.

Android Application

In the Android Application that we have built there is a basic GUI which has a joystick using which we can easily tell in which direction do we have to move on.



GUI of the API

CODE

```
#define REMOTEXY_MODE__SOFTWARESERIAL
#include <SoftwareSerial.h>              //Including the software serial library
#include <RemoteXY.h>                    //Including the remotexy library

/* RemoteXY connection settings */
#define REMOTEXY_SERIAL_RX 0            //defining the pin 2 as RX pin
#define REMOTEXY_SERIAL_TX 1             //defining the pin 3 as TX pin
#define REMOTEXY_SERIAL_SPEED 9600       //setting baudrate at 9600

unsigned char RemoteXY_CONF[] =        //remotexy configuration
  { 3,0,23,0,1,5,5,15,41,11
  ,43,43,1,2,0,6,5,27,11,5
  ,79,78,0,79,70,70,0 };

struct {                           //Function for declaring the variables
  signed char joystick_1_x;        //joystick x-axis
  signed char joystick_1_y;        //joystick y-axis
  unsigned char switch_1;          //variables for switch
  unsigned char connect_flag;

} RemoteXY;
```

```
//defining the pins for first motor
#define IN1 2
#define IN2 4

//defining the pins for second motor
#define IN3 5
#define IN4 6


//defining the LED pin
#define ledpin 13

unsigned char first_motor[3] =
  {IN1, IN2};
unsigned char second_motor[3] =
  {IN3, IN4};

void Speed (unsigned char * pointer, int motor_speed)
{
  if (motor_speed>100) motor_speed=100;
  if (motor_speed<-100) motor_speed=-100;
  if (motor_speed>0) {
```

```
unsigned char first_motor[3] =
  {IN1, IN2};
unsigned char second_motor[3] =
  {IN3, IN4};

void Speed (unsigned char * pointer, int motor_speed)
{
  if (motor_speed>100) motor_speed=100;
  if (motor_speed<-100) motor_speed=-100;
  if (motor_speed>0) {
    digitalWrite(pointer[0], HIGH);
    digitalWrite(pointer[1], LOW);

  }
  else if (motor_speed<0) {
    digitalWrite(pointer[0], LOW);
    digitalWrite(pointer[1], HIGH);

  }
  else {
    digitalWrite(pointer[0], LOW);
    digitalWrite(pointer[1], LOW);

  }
}

void setup()
{
  //defining the motor pins as the output pins
  pinMode (IN1, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (IN4, OUTPUT);
  pinMode (ledpin, OUTPUT);
  RemoteXY_Init ();
}

void loop()
{
  RemoteXY_Handler ();
  digitalWrite (ledpin, (RemoteXY.switch_1==0)?LOW:HIGH);
  Speed (first_motor, RemoteXY.joystick_1_y - RemoteXY.joystick_1_x);
  Speed (second_motor, RemoteXY.joystick_1_y + RemoteXY.joystick_1_x);
}
```

RADAR MAPPING BLOCK

Collection of Data FOR 3D MAPPING:

- Data will be collected using a HCSR04 Ultrasonic sensor controlled by an arduino and then will be transmitted to the PC using serial communication.
- Servo motor will rotate the ultrasonic sensor to give 180 angle view
- The data received will then be stored in the form of two variables and then will be used in a function to generate a signal.
- A Graphic User Interface will be developed on processing on which this data will be plotted.
- The digital signal acquired from the arduino will be transmitted using Bluetooth communication to the laptop which is now processed and converted into an analog image format to be displayed on the GUI.

Signal processing:

The signal is sent to the obstacle in for of pulse and any disturbance in the pulse is observed as a obstruction. Distance from the place of obstacle is then calculated as:

distance= duration*0.034/2

This gives us the distance of the object at a radial line and then this data is processed and plotted on a radial graph.

Plotting the Graph:

- Our ultrasonic sensor can measure distance within a range of 2 cm to 4 m. We also use a Servo motor to actuate movements, mainly giving the robot arm its precise angle.
- Since the servo swings in an arc, we can think about the output of the servo motor as an angle.
- As a test experiment, we will try taking a smaller range between 0 and 7 inches.
- The range we had chosen for our test sample is 0 to 7 inches.
- We have to place a pointer on the servo and have it point at a scale that will indicate distance.

The scale we have chosen for this is to have the range of the angles between 37 degrees and 143 degrees. For a distance measured of 0 inches, we want the servo motor to point at 37 degrees. For a distance measured of 7 inches, we want it to point at 143 degrees.

GUI Using Processing JAVA:

Processing is an open source programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks. We send the data of angle and the obstacle distance to the laptop via serial communication and using this software we map the entire data on the GUI developed.

Arduino code for mapping

```
// Includes the Servo library
#include<Servo.h>
#include <SoftwareSerial.h>
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 4;
const int echoPin = 2;
SoftwareSerial BTserial(0,1);
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  BTserial.begin(9600);
  myServo.attach(6); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(10);
  distance = calculateDistance();// Calls a function for calculating the distanc

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous valu
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous valu
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(10);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 // Sets the trigPin on HIGH state for 10 micro seconds
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound
 distance= duration*0.034/2;
 return distance;
}
```

Processing code to display the data:

```
import processing.serial.*; // imports library for serial comm
import java.awt.event.KeyEvent; // imports library for reading
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (1200, 700); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION
 smooth();
 myPort = new Serial(this,"COM13", 9600); // starts the serial
 myPort.bufferUntil('.'); // reads the data from the serial pc
}
void draw() {

  fill(98,245,31);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();

void serialEvent (Serial myPort) { // starts reading data from the Serial Por
  // reads the data from the Serial Port up to the character '.' and puts it
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the
  angle= data.substring(0, index1); // read the data from position "0" to pos
  distance= data.substring(index1+1, data.length()); // read the data from pc

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
```

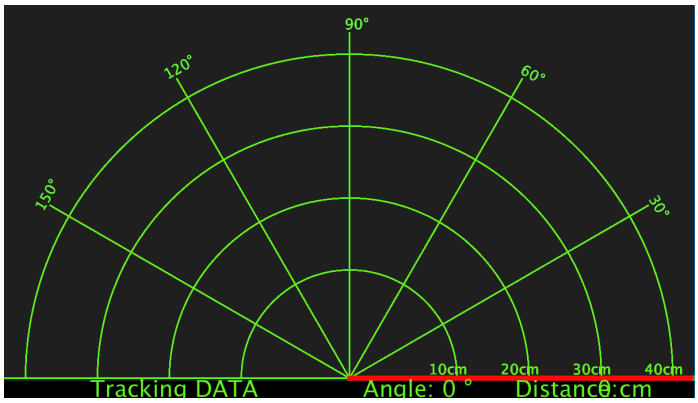
```
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new locat
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),
  (width-width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new locat
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)
  *sin(radians(iAngle))); // draws the line according to the angle
  popMatrix();
}
void drawText() { // draws the texts on the screen

  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
   else {
   noObject = "In Range";
   }
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("1cm",width-width*0.3854,height-height*0.0833);
  text("2cm",width-width*0.281,height-height*0.0833);
  text("3cm",width-width*0.177,height-height*0.0833);
  text("4cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Tracking DATA ", width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("        " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*sin(ra
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*sin(rad
  rotate(-radians(-30));
```

```
text("60°",0,0);
resetMatrix();
translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*sin(rad
rotate(radians(0));
text("90°",0,0);
resetMatrix();
translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*sin(rad
rotate(radians(-30));
text("120°",0,0);
resetMatrix();
translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2*sir
rotate(radians(-60));
text("150°",0,0);
popMatrix();
```

GUI of the RADAR MAPPING



Video Surveillance Block:

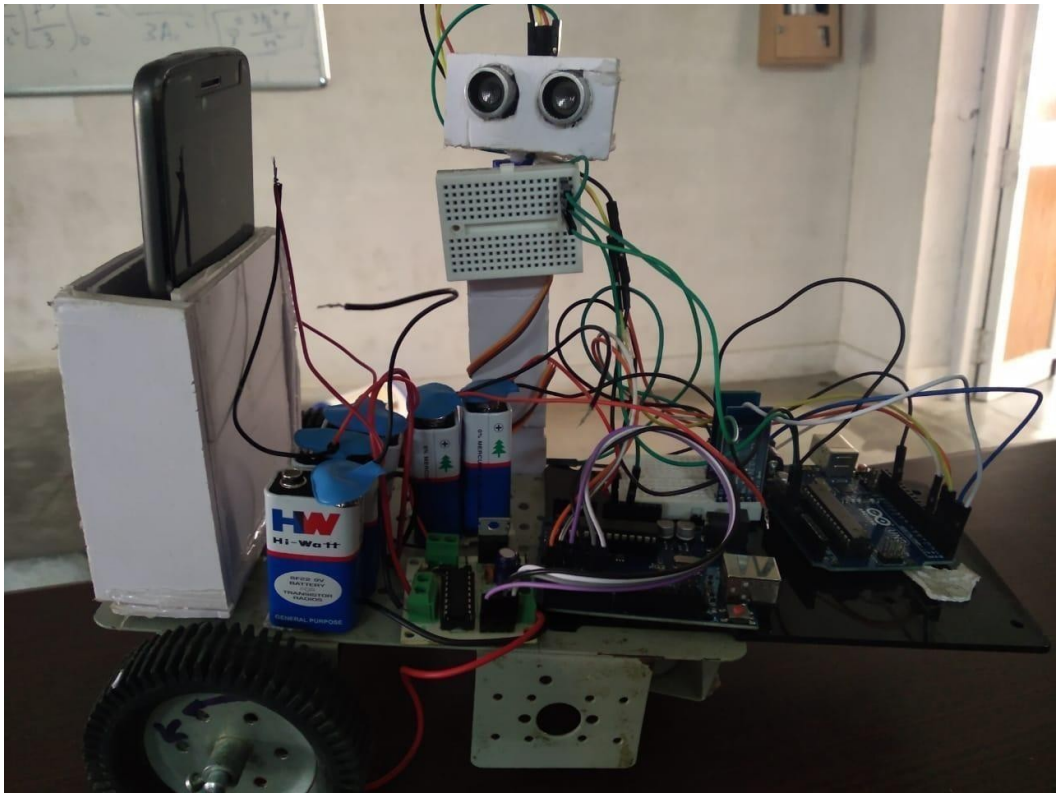
- For video surveillance we are using a mobile camera. In the mobile phone we have installed a mobile application which takes the video feed and generates a server using the wifi of the mobile phone. Gradually it renders the video on the server.
- Our base station is then connected to the server. Using a webpage whose code is written in Java script we connect the laptop to the server and download all the video feed and display it on any web browser.

Code

```
Elements  Console  Sources  Network  »
<html> == $0
  <head>
    <title>Android Webcam Server</title>
    <link href="style.css" type="text/css" rel="stylesheet">
    <script type="text/javascript" src="js/jquerymin.js"></script>
    <script type="text/javascript" src="js/ipwebcam.js"></script>
    <style type="text/css">
    body { margin: 0 }
    </style>
    <script type="text/javascript">
    $(loadBrowserFullscreen);
    </script>
  </head>
  <body>
    <img id="img1" onclick="swBAspect()" src="/video" alt="Video feed"
    style="height: 657px;">
  </body>
</html>
```

PROGRESS COMPLETE

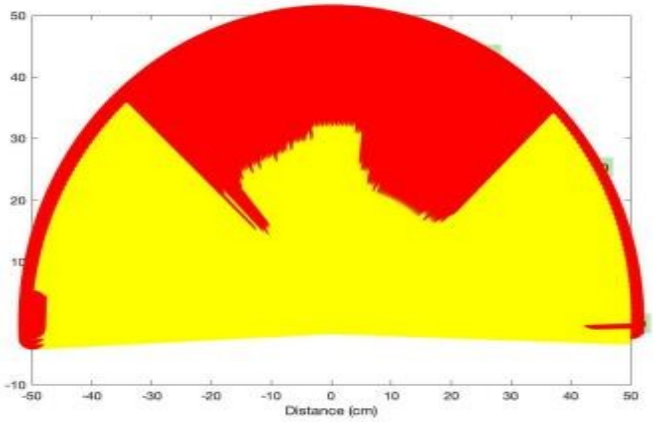
90% of our project was complete before the lockdown. We had developed a working prototype of the model and tested it in inner surroundings. The remaining part to be done was to test our model in external surroundings to see how it reacts to various obstacles.



Final setup of the BOT

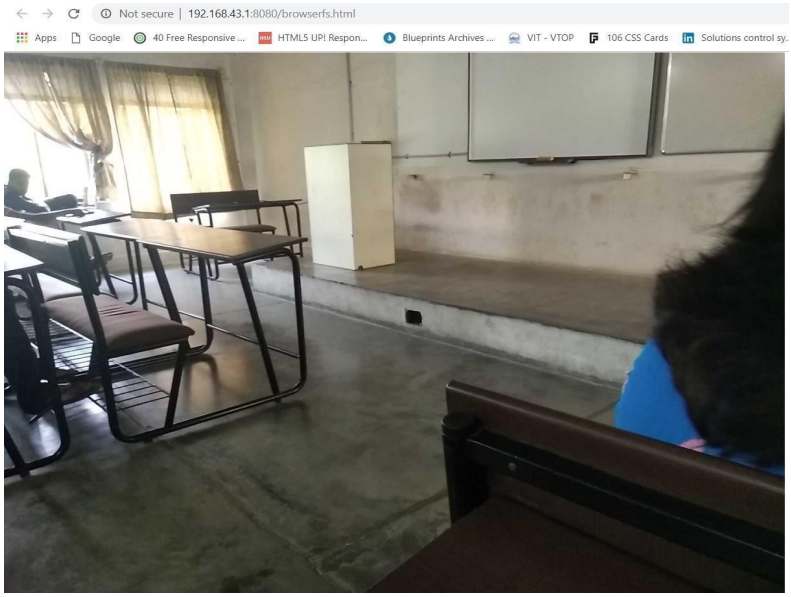
By using ultrasonic sensor we were able to map the obstacles in the location accurately and with the use of smart phone we also did get the live surveillance. Together those two make an intelligent device. Whenever an obstacle is detected using ultrasonic sensor the wheels are adjusted in such a way that the vehicle evades the obstacle and the streaming from the smart phone is focused on other areas.

Ultrasonic Radar Framework result



The yellow part shows that there is no deflection or no object is in the path and red part shows that a deflection has occurred due to an object in a particular region, utilizing this we can likewise trace the position and angle of the object.

Video Surveillance



CONCLUSION

Our project presented the importance of visual surveillance of unknown territories and analysing them before any human intervention. The system is configured using an Arduino, ultrasonic sensor that detects obstacles on its path and can also be used for intrusion detection for location sizes. While servomotor is responsible for the mechanical motion, the RADAR subsystem simultaneously plots the deflection onto the ultrasonic radar framework. Further work is related to the consideration of enhancing the surveillance system to not only transmit the live stream but also simultaneously analyse the environment that it is in to detect the toxicity levels.

REFERENCES

[1] Tedeschi, A., Calcaterra,S. and Benedetto, F, “Ultrasonic RAdar system (URAS): Arduino and virtual reality for a light-free mapping of indoor environments”, IEEE Sensors Journal, 2017,17(14), pp.4595-4604.

[2] Aditya Chaudhry, Manas Batra, Prakhar Gupta, Sahil Lamba, Suyash Gupta, “Arduino based voice controlled robot”, International Conference on Computing,
Communication, and Intelligent Systems (ICCCIS), 2019

[3] Hou-Tsan Lee, Wei-Chuan Lin, Ching-Hsiang Huang, Yu-Jhih Huang, “Wireless Indoor Surveillance Robot”, SICE Annual Conference, Waseda University, Tokyo, Japan,2011

[4] D. B. Kadam, Y. B. Patil, K. V. Chougale and. S. S. Perdeshi, "Arduino Based Moving Radar System", International Journal of Innovative Studies in Sciences and Engineering Technology (IJISSET), vol. 3, no. 4, pp. 23-27,2017.