

SAFECAGE **Safety Assistance Surveillance Bot**

Project Report

Submitted for the course:

Robotics and Automation

ECE2008

E1 Slot

Winter Semester 2018-19

Manasa R	17BEC0409
Ranjith DVL	16BEC0550
Souvik Dandapat	16BEC0589
Anurag Prashar	16BEC0235

Submitted to

Prof. Abraham Sampson S
Assistant Professor (Senior)

School of Electronics Engineering



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

ACKNOWLEDGEMENTS

We would like to thank our professor Prof. Abraham Sampson S. whom we are highly indebted to for giving us this opportunity to perform this project and for his help, and guidance throughout this project.

We would like to express our gratitude to VIT University and the School of Electronics and Communication Engineering (SENSE). We would like to extend our gratitude to the Dean of SENSE and this prestigious University in supporting us and giving us an opportunity to carry out our studies at this University. And finally, we would like to thank our Parents and my friends for supporting and helping us throughout our project.

Declaration by Author(s)

Ranjith DVL

BTECH, ECE (School of Electronics Engineering), 3rd year, VIT, Vellore

Souvik Dandapat

BTECH, ECE (School of Electronics Engineering), 3rd year, VIT, Vellore

Manasa R

BTECH, ECE (School of Electronics Engineering), 2nd year, VIT, Vellore

Anurag Prashar

BTECH, ECE (School of Electronics Engineering), 3rd year, VIT, Vellore

Contents:

• Abstract	4
• Introduction	5
• Methodology	6
• Technical blocks	7
• Components required	7
• Block diagrams	9
• Working	10
• Android App	12
• The Code	13
• Radar Mapping	15
• Applications	23
• Conclusions	24
• References	25

Abstract

Security aspects are the main focus of the project. In the home environment, it becomes more important to provide security from thefts and other unusual problems. With this project, it becomes possible to monitor the surrounding environment of the house, move the robot in the house, set up the alarm whenever intrusion occurs, when the gas leakage occurs and the signaling of temperature variation. Silent alarm and alarm signal detect the surrounding intrusions.

Robots can be of great use to humans. They can be regarded as fruitful substitutes to watchmen. In this way, they provide security from thefts, robberies and unusual occurrences at home too. This Human-Robot application is easy to build and deploy on any far machine too.

There is a need for high surveillance especially in sensitive areas, military borders, government offices, public places, security check points and even in the homes. Country borders, public places, government offices, etc comes in the category of outdoor surveillance while warehouses, homes, garages fall in the category of indoor surveillance. CCTV and other cameras have their own drawbacks and constraints such as they usually have blind spots and they can be covered with sticking something over the lenses like gum or spray paint, also to cover the entire area many cameras are required to reach every nook and corner increasing the cost. Similarly human beings also have their own limitations such as obtaining information from inaccessible places is not possible, at the same time human life is precious. Thus humans cannot always be used to acquire information from remote and dangerous places. A robot capable of moving in all types of terrains and mounted with camera and other sensors can do the job. It can help in outdoor surveillance by monitoring important places and also it can be made to reach suspicious place for close observations.

Introduction

Disasters can be classified as natural and man-made disasters. Natural disasters include cyclones, floods, storms, volcanic eruptions, earthquakes, Tsunami, hailstorms, wildfire, etc. Man-made disasters are nuclear power plant accidents, war, minefields, oil spills, radiation, hijacks, etc. Such disasters strike countries causing huge destruction and human deaths in the number of millions. During such situations to save lives of people numerous essential services are deployed.

Rescue involves responsive operations that involves saving the life of people stuck in a dangerous situation. In most of the cases rescue team is unable to enter in some difficult parts of affected area. Even if the team is able to reach the people it is dangerous and risky to enter such areas without getting the prior information about the number of victims, their condition and environmental conditions of the area. Rescuing people from a hijack situation also requires a detailed information about the positions of hostages and terrorists so that the rescue members can prepare themselves. Sometimes due to lack of information a rescuer may become a victim.

In such risky and dangerous situations 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. The robot should be able to provide live video of the situation, detect human presence and other environmental related information such as humidity, temperature, presence of smoke, geographic location in terms of latitude and longitude.

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 Smart phone 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.

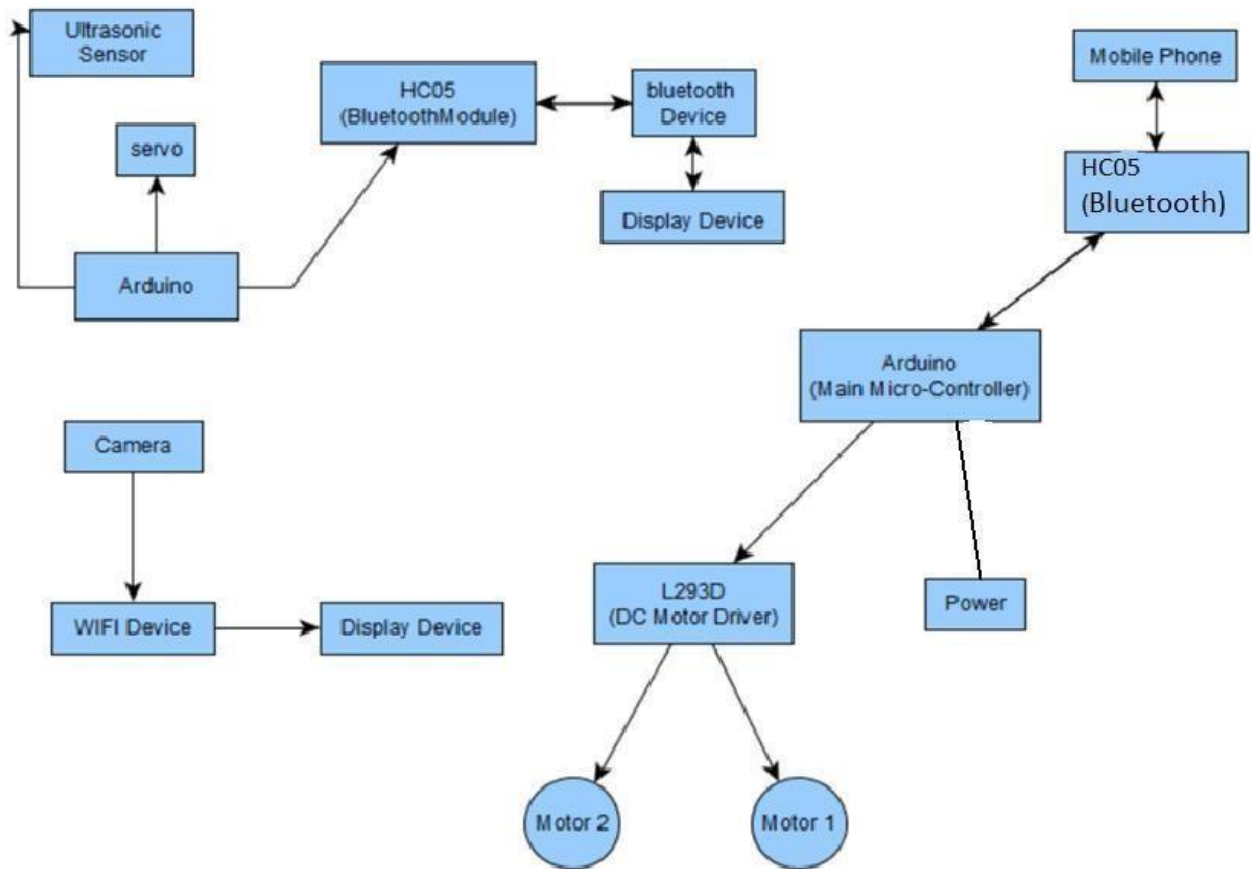
1. **Arduino Uno:** Arduino Uno is a microcontroller board based on the ATmega328P It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. Simply connect it to a computer with a USB cable or power it with a AC- to- DC adapter or battery "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0.
2. **L293D Motor Driver** L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively. When an enable input is high, the associated driver gets enabled.

3. **HS CR 05 Ultrasonic Sensor:** Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The basic principle of work:
 - Using IO trigger for at least 10us high level signal.
 - The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
 - If the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.
4. **HC05 Bluetooth Module:** HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH.
5. **Servo Motor:** A 9g servo to rotate the ultrasonic to give 180 degrees of data
6. **GPS Module:** GPS module used in the mobile phones
7. **Camera Module:** Camera present in the mobile phones

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 L293D programmed 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 server by using Blank free server
5. Processing is used to map the data obtained from the ultrasonic sensor

Block Diagram



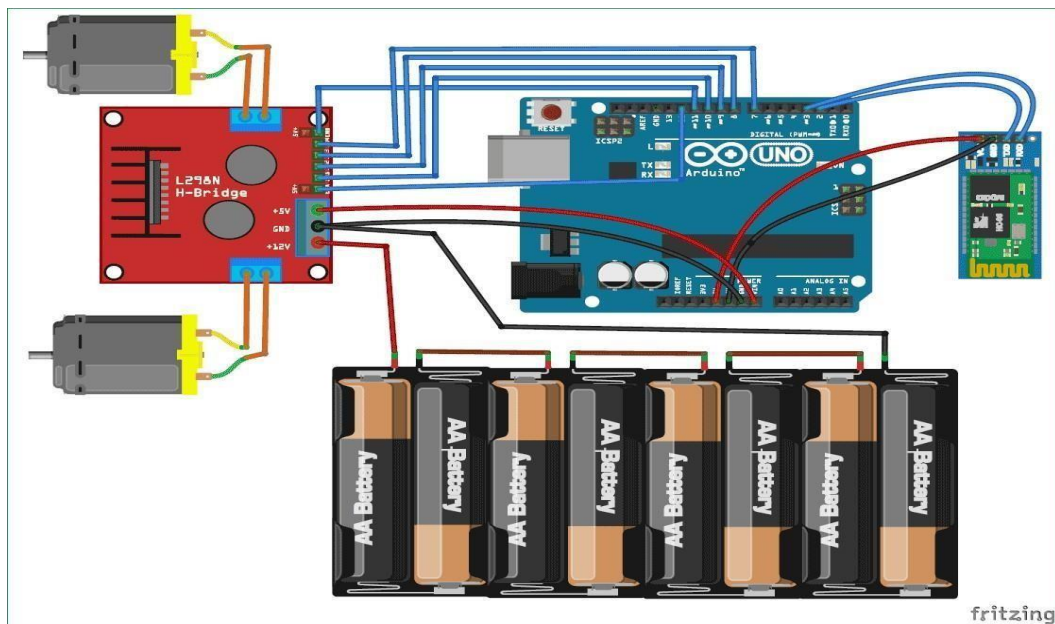
Working:

➤ 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.

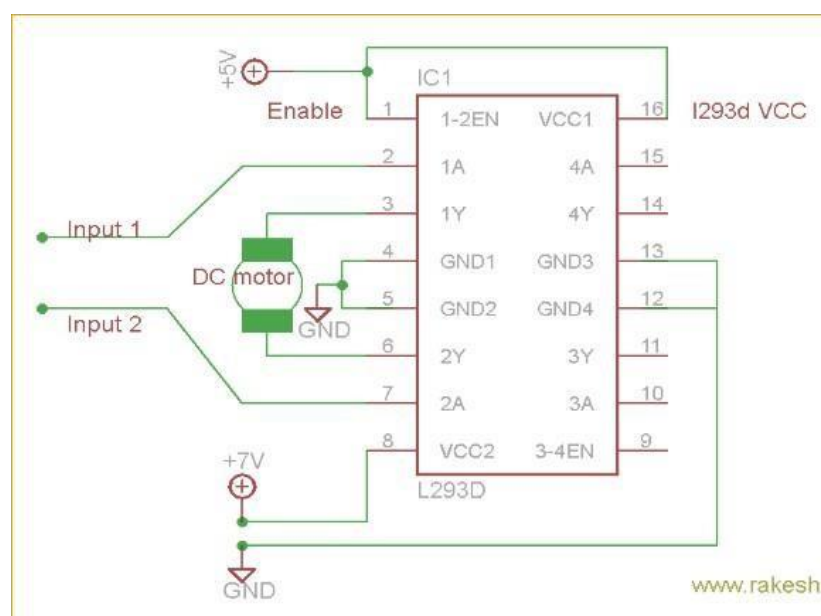
L293D Logic Table.

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 [No rotation]

So to move in a particular direction:

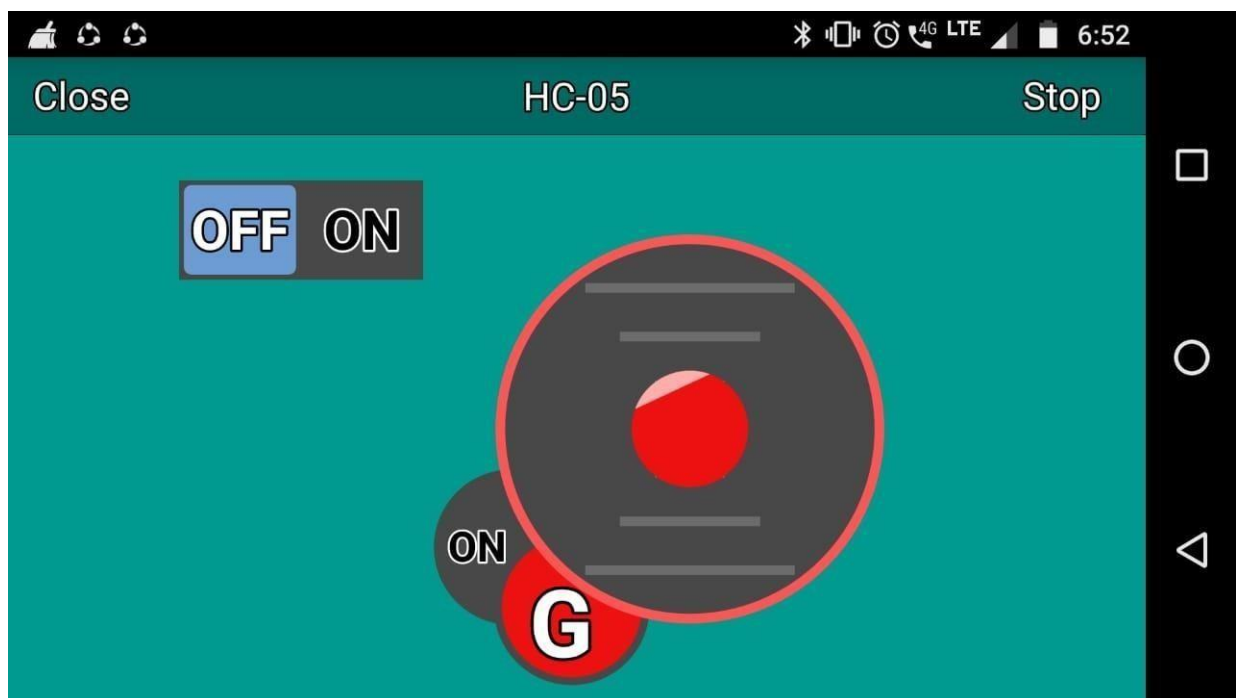
Direction	Left motor	Right motor
Forward	Clockwise	Clockwise
Backward	Anti-clockwise	Anti-clockwise
Left	Clockwise	Anti-clockwise
Right	Anti-clockwise	Clockwise



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.

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



GUI of the API

The 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_l_x;         //joystick x-axis
    signed char joystick_l_y;         //joystick y-axis
    unsigned char switch_l;          //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);
}

```

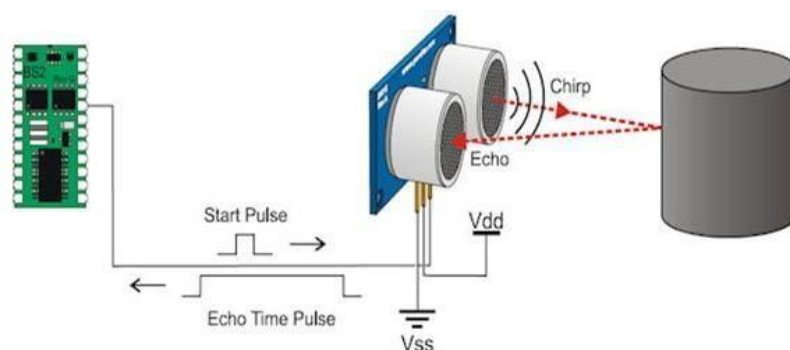
Radar Mapping Block

Collection of Data FOR 3D MAPPING:

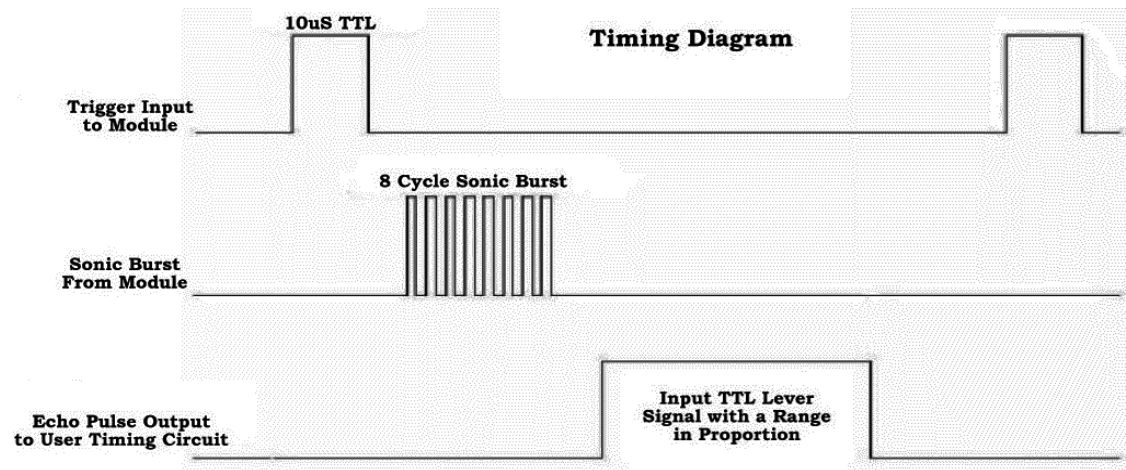
- Data will be collected using a HCSR04 Ultrasonic sensor controlled by a arduino and then will be transmitted to the PC using serialcommunication.
- Servo motor will rotate the ultrasonic sensor to give 180 angle view
- The data received will then be stored in the form of two variables and then will 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 transimitted using Bluetooth communication to the laptop which is now processed and converted into an analog image format to be displayed on the GUI.

Principle of Operation of Ultrasonic Sensor:

- The Arduino generates a short 10uS pulse to the Trigger input to start the ranging. The Ultrasonic Module will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high.
- It then listens for an echo, and as soon as it detects one it lowers the echo line again.
- The echo line is nothing but a pulse whose width is proportional to the distance to the object.



Diagrammatic Representation:



Signal processing:

The signal is sent to the obstacle in form of pulse and any disturbance in the pulse is observed as an obstruction.

Distance from the place of obstacle is then calculated as:

$$\text{distance} = \text{duration} \times 0.034 / 2$$

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 Trig 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 distance

    Serial.print(i); // Sends the current degree into the Serial Port
    Serial.print(","); // Sends addition character right next to the previous value
    Serial.print(distance); // Sends the distance value into the Serial Port
    Serial.print("."); // Sends addition character right next to the previous value
  }
  // 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(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 Port
  // 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 po

  // 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 local
  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 local
  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";
  }
}

```

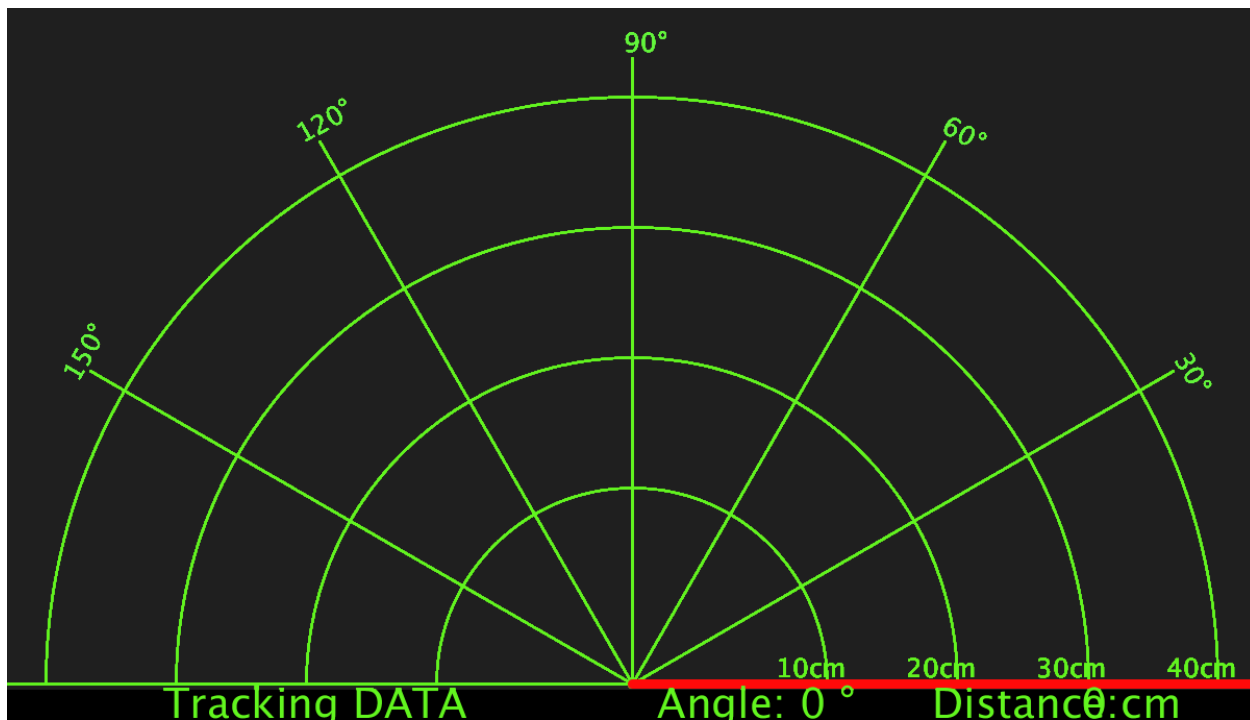
```

    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(rad
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*sin
rotate(radians(-60)));
text("150°",0,0);
popMatrix();

```

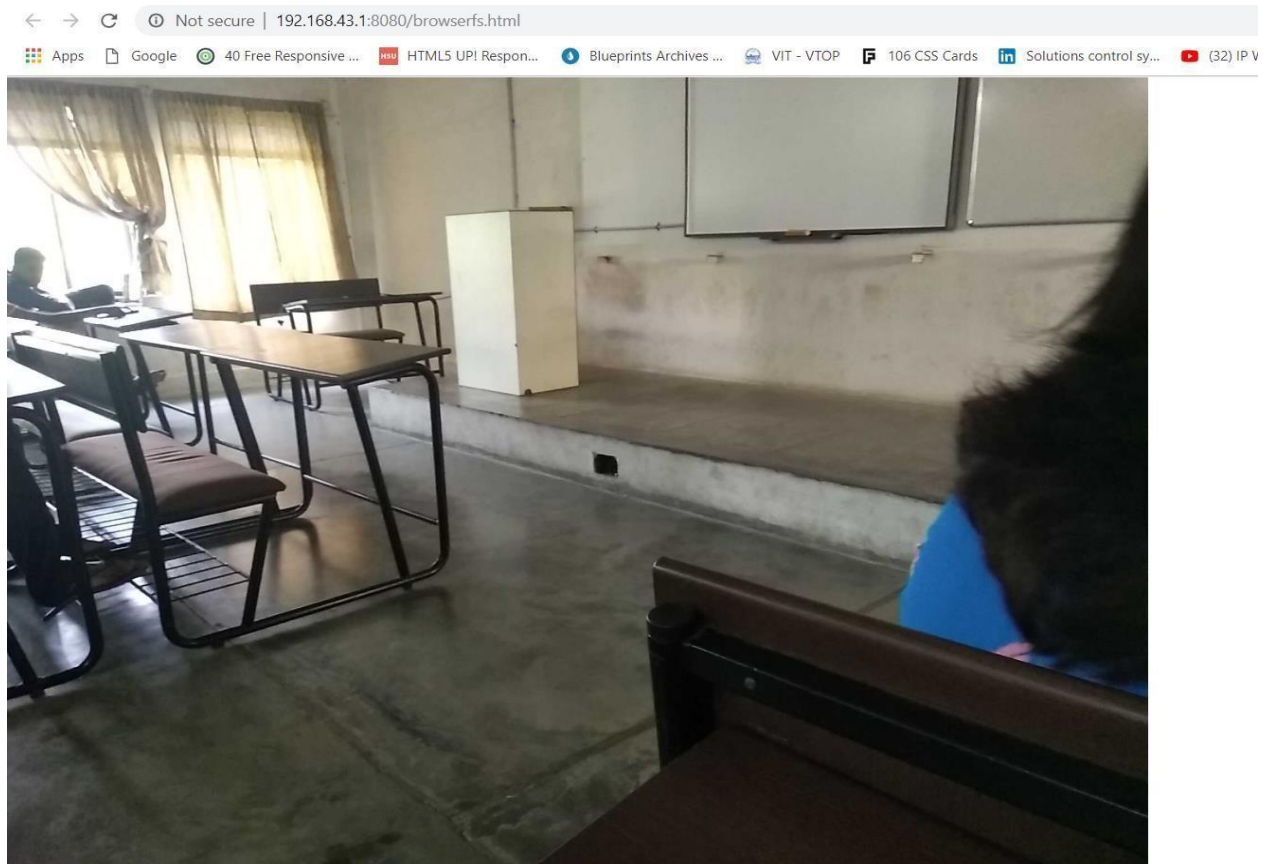
GUI of the RADAR MAPPING



Video Surveillance Block :

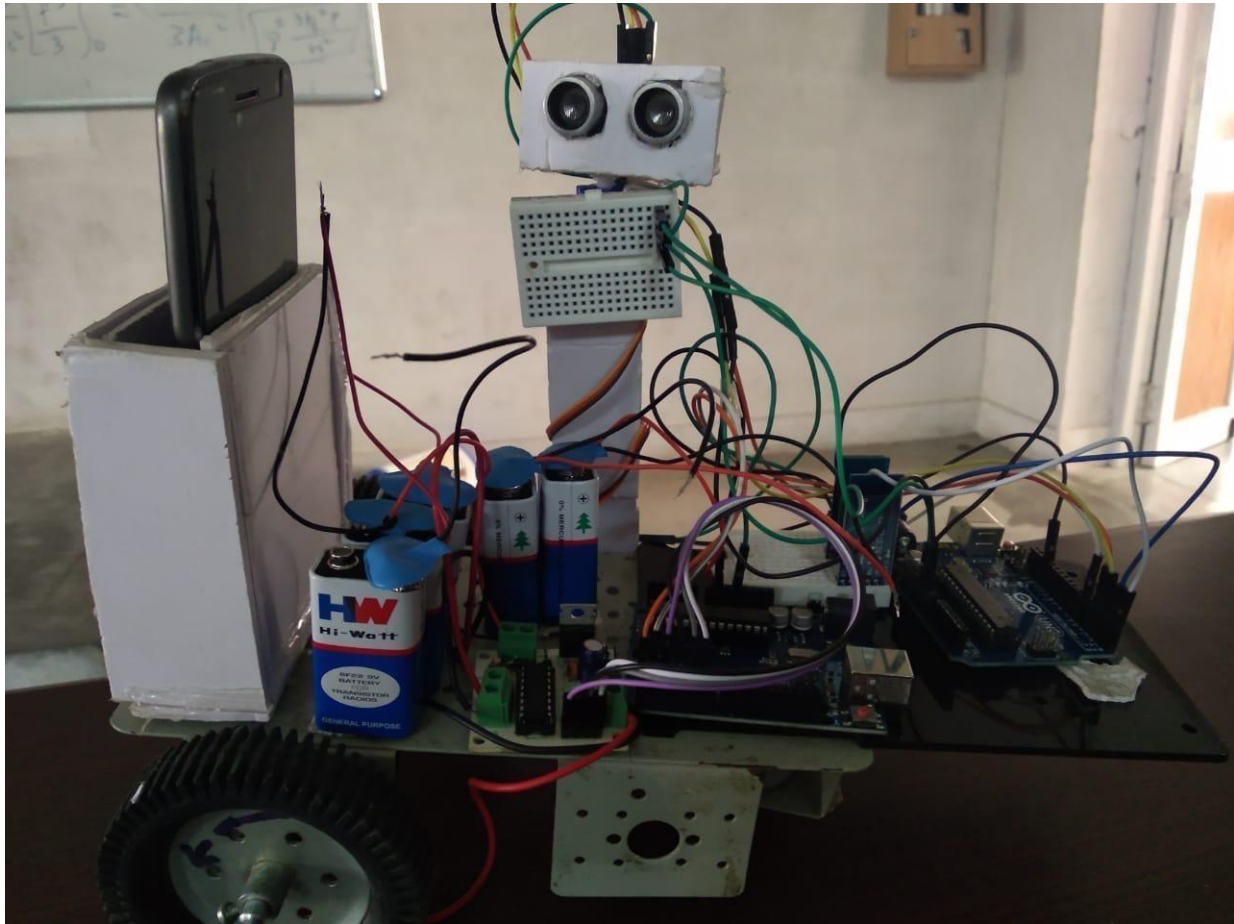
For video surveillance we are using a mobile camera. In the mobile phone we have installed a mobile application which takes the video feed and generates a server using the wifi of the mobile phone. Gradually it renders the video on the server.

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>
  
</body>
</html>
```



Final setup of the BOT

Applications:

- This Bot can be used as a surveillance device that can be used to for spying purposes
- Used for surveillance in malls and campuses.
- Used in to map the surroundings of a region that is physicalinaccessible, for example a cave.
- Used to enter region composed of high radioactivity that are dangerous for humans to analyse and gather data.

Conclusion:

The project made above has already been tested and utilized in various areas of expertizes like:

1. Used as Stealth Surveillance Robot
2. Used to scan mines and caves inaccessible to Human body
3. Used to Defuse Bombs
4. Used in areas with High Radioactive Radiations

References:

1. Remote control robot using Android mobile device

Published in: Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)

Date of Conference: 28-30 May 2014

Date Added to IEEE Xplore: 26 June 2014

ISBN Information:

INSPEC Accession Number: 14400043

DOI: 10.1109/CarpathianCC.2014.6843630

Publisher: IEEE

Conference Location: Velke Karlovice, Czech Republic

The paper describes the design and realization of the mobile application for the Android operating system which is focused on manual control of mobile robot using wireless Bluetooth technology. The application allows the robot control interaction with the display, or voice. When we use a graphical interface, we can monitor the current distance of the robot from obstacles. The measurement of distance is carried out by ultrasonic sensor placed in front of the robot. It was necessary to build a prototype of a mobile robot for the development of the

application. The prototype of the mobile robot is based on the differential gear.

2. Object recognition with ultrasonic sensors

Published in: Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)

Date of Conference: 8-9 Nov. 1999

Date Added to IEEE *Xplore*: 06 August 2002

Print ISBN: 0-7803-5806-6

INSPEC Accession Number: 6461515

DOI: 10.1109/CIRA.1999.810057

Publisher: IEEE

Conference Location: Monterey, CA, USA, USA

We have recently introduced a novel approach to sonar based object recognition for robotics. The sonar recognition system, consisting of a Polaroid sonar coupled with art A/D data acquisition board in a LINUX-based PC, uses a fuzzy ARTMAP neural network classification system to recognize objects at varying distances. In this article we report additional results in which we test systematically recognition performance using different kinds of information from the sonar echo, and different object configurations. The results strengthen our claims that sonar can be used as a viable system for real-time object recognition in robotics and other application domains.

3. Video surveillance robot control using smartphone and Raspberry pi

Published in: 2016 International Conference on Communication and Signal Processing (ICCSP)

Date of Conference: 6-8 April 2016

Date Added to IEEE *Xplore*: 24 November 2016

ISBN Information:

INSPEC Accession Number: 16488791

DOI: 10.1109/ICCSP.2016.7754547

Publisher: IEEE

Conference Location: Melmaruvathur, India

This paper proposes a method for controlling a wireless robot for surveillance using an application built on Android platform. The Android application will open a web-page which has video screen for surveillance and buttons to control robot and camera. Android Smartphone and Raspberry pi board is connected to Wi-Fi. An Android Smartphone sends a wireless command which is received by Raspberry pi board and accordingly robot moves. The Video Streaming is done using MJPG streamer program that gets mjpeg data and sends it through a HTTP session. The Raspberry pi programming is done in python language. The experimental result shows that the video streamed up to 15 frames per second.

4. Development of a Wireless Surveillance Robot for Controlling from Long Distance

International Journal of Engineering Research And Management (IJERM)
ISSN: 2349- 2058, Volume-05, Issue-09, September 2018

A robot has been developed which can be used for multipurpose application related to surveillance and security systems. From ground testing of this robot it has been found that it can be controlled from unlimited distance as the system is based on World Wide Web (www). This robot also has a teleoperation system based on radio frequency (RF) for signal processing. It has been found that this robot shows about 78% efficiency when a constant 512 kbps Wi-Fi internet connection is applied. Visual Basic software has been used to operate the robot. Four cameras are attached with the robot for acquisition of images from the surroundings. Vmcap software is used to show all these camera images at a time. The images from the four cameras and the control panel will be transmitted and displayed in the user's monitor. Virtual network computing (VNC) software has been employed for the purpose. By watching the situational images sent by the remote robot it is the user who can control the motion as well as can get the idea of the environment. A GPS (Global Positioning system) device is mounted over the robot to get the satellite image of the mobile robot trajectory of motion and a total track map. This paper describes the methods and scopes of the above-mentioned robot.