## OBSTACLE DETECTION SYSTEM

A Minor Project work submitted in partial fulfilment of the requirement for the award of the degree of

### BACHELOR OF TECHNOLOGY

**in**

### ELECTRONICS & COMMUNICATION ENGINEERING

**By**

### N.GUNA SEKHAR(19211A04E2)

### M.MAHESHKUMAR(19211A04C8)

**P.AKASH VARMA (19211A04H4)**

**Under the esteemed guidance of**

**Mr.B.ANIL KUMAR**

## B.V. Raju Institute of Technology

### UGC- AUTONOMOUS

**Department of Electronics and Communication Engineering Vishnupur, Narsapur, Medak. (Dst) (Affiliated to JNTU, Hyderabad)**

**2020 – 2021**

## I

**B. V. Raju Institute of Technology**

#### Vishnupur,Narsapur, Medak(Dt). Pin: 502313 (Affiliated to JNTU, Hyderabad)

**Ph.:08458-222000,222001, Fax:08458-222002**

**Department of Electronics & Communication Engineering**



CERTIFICATE

This is to certify that the Minor Project work entitled Object Detection System is being submitted by N.Guna sekhar (19211A04E2), M.Mahesh (19211A04C8) and P.Akash varma (19211A04H4) in partial fulfillment of the requirement for the award of the degree of **B.Tech.in Electronics & Communication Engineering**, by Jawaharlal Nehru Technological University Hyderabad is a record of bonafide work carried out by them under my guidance and supervision from 2020 to 2021.

The results presented in this project have been verified and are found to be satisfactory.

#### INTERNAL GUIDE HEAD OF THE DEPARTMENT

#### Dr.Sanjay Dubey

**Mr.B.ANIL KUMAR PhD, Mtech, MISTE,MIEEE. Assistant Professor,M.Tech, Professor & HOD, Dept. of ECE**

**PhD**

**EXTERNAL EXAMINER**

#### II



Department of Electronics and Communication

Engineering

## ROBOTICS CENTRE FOR ENHANCED LEARNING

CERTIFICATE

This is to certify that Mr.N.Guna sekhar, Mr M.Mahesh, Mr. P.Akash varma bearing Roll Nos. 19211A04E2, 19211A04C8, 19211A04H4 respectively have successfully completed his/her training on Robotics and Implemented a Project titled **“ Object Detection and Tracking using Open CV,Visual Studio C++2010 and Arduino ”** in Centre for Robotics Centre for enhanced learning Laboratory, B.V. Raju Institute of Technology from **2019** to **2023**.

#### Head of the Department

**Dr. Sanjay Dubey**

Ph.D.,Post.Doc

MISTE, MIEEE.

**III**

## ACKNOWLEDGEMENT

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

Foremost I would like to offer this endeavour to our GOD almighty for the wisdom he bestowed upon me, the strength, the peace of my mind and good health in order to finish this research.

We are grateful towards our College Management and our beloved Principal **Dr.**

**K. Lakshmi Prasad** for providing us the necessary infrastructure and facilities that ensured smooth and satisfactory execution of the project.

We would like to express out profound gratitude to our Head of the department **Dr. Sanjay Dubey, Professor & HOD**, Dept. of ECE, for his encouragement inspiration and close monitoring and guidance he gave us during the execution of the project.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to our faculty co-ordinator, mentor **B.Anil kumar** and team for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & members of BVRIT for their kind co-operation and encouragement which help me in completion of this project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

### Names

1. **N.Guna sekhar 19211A04E2**

### M.Mahesh kumar 19211A04C8

1. **P.Akash varma 19211A04H4**

## IV

## DECLARATION

We hereby declare that the project entitled**“OBSTACLE DETECTION AND TRACKING USING C++ AND AURDINO**“ submitted to B. V. Raju Institute of technology, affiliated to Jawaharlal Nehru Technological University, Hyderabad for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering is a result of original project work done by us.

It is further declared that the project report on any part therefore has not been previously submitted to any University or Institute for the award of degree or diploma.

## V

### ABSTRACT

1. the object In this project the Webcam sends video frames to the Visual Studio C++ which contains Open CV library running on our computer.If Visual Studio C++ program detects the image of the object from the webcam then it calculates the co ordinates of X, Y axis and radius of the object. The co ordinates are sent accordingly to the arduino UNO via Serial communication between the arduino and visual studio c++.After receiving the co-ordinates the servo motors moves in X and Y direction and follows.

## VI PREFACE

As a part of the B. tech curriculum and in order to gain practical knowledge in the field of Electronics and Communication, we are required to make a project report on “**obstacle detector using image processing**”. The report is prepared with the view to include all the details regarding the project that I carried out.

In this project we have included various concepts, technology and implementation regarding Object detection. Subject to the limitation of time efforts and resources every possible attempt has been made to study the problem deeply.

Doing this project report helped us to enhance our knowledge regarding the work. Through this report we come to know about importance of teamwork and role of devotion towards the work.

## VII

## CONTENTS

CERTIFICATE I

SPECIAL LAB CERTIFICATE II

[ACKNOWLEDGEMENT III](#_TOC_250006)

DECLARATION IV

[ABSTRACT V](#_TOC_250005)

PREFACE VI

[CONTENTS VII](#_TOC_250004)

LIST OF FIGURES & TABLES VIII

1. INTRODUCTION 1-3
   1. INTRODUCTION TO OBSTACLE DETECTOR WITH 2

IMAGE PROCESSING

* 1. OBJECTIVE 2
  2. MOTIVATION 3

1. LITERATURE SURVEY 4-5
   1. INTRODUCTION 5
2. ANALYSIS AND DESIGN 6-13
   1. Components used 7-11
   2. SOFTWARE TOOLS USED 12-13
   3. BLOCK DIAGRAM 14
3. IMPLEMENTATION 14-21
   1. WORKING 15
   2. ARDUINO CODE 17-19
   3. VISUAL STUDIO CODE 19-27
4. RESULTS 29-30
5. APPLICATIONS, ADVANTAGES, LIMITATIONS 31-34
   1. APPLICATIONS 32
   2. ADVANTAGES 33
   3. LIMITATIONS 34
6. CONCLUSION 35-36
   1. CONCLUSION 35
   2. FUTURE SCOPE 36
7. REFERENCE 37

**VIII**

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **Figure No.** | **Name of the figure** | **Page No.** |
| 3.1 | Arduino Uno | 7 |
| 3.2 | Servo Motor | 9 |
| 3.3 | Bread Board | 10 |
| 3.4 | Webcam | 11 |
| 3.5 | Arduino IDE | 12 |
| 3.6 | Block Diagram | 13 |
| 3.7 | Circuit Daigram | 14 |
| 5.1 | Hardware connections | 29 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# CHAPTER 1 INTRODUCTION

**1**

### Introduction to project:

Contactless distance measurement can be done by many different methods, according to the selected principle. In various applications, transmitters and receivers use infrared light, the distance being measured by the method of optical triangulation.

Motion detection of objects or living beings might be interesting in many domains, such as security devices, the positioning of industrial robots, liquid levels in tanks, the depth of snowbanks and auto guidance systems. Most of these applications require that the detection system to be non-invasive and to not disturb the normal work environment, devices or living beings in the area of detection. This involve the choice of discrete vectors for information transport, with high immunity to all other factors unless the state of motion.

The obstacle detection field is a very broad one and a lot of obstacle detection systems have been developed in the last years in this domain.

The project developed by us has a wide range of applications. Be it for security purposes or road safety. Nowadays most of the people face problem while parking in crowded areas. With the help of this device, the driver can detect can object while parking.

This project can also be extended for security purposes. For example, we come across situations where we need to keep a watch over prohibited areas to avoid trespassing. Now keeping human labour for this purpose is costly and not reliable for keeping a watch over an area 24×7.So this project can also be used for unauthorized human / animal or object detection system. The system can monitor an area of limited range and alerts authorities with a buzzer as an alarm.

### Motivation:

The motivation behind doing this project is that, something that will over do all the physical tasks. Robotics and smart systems are buzzing around all over the world. Object recognition and tracking reduces human efforts and provides efficiency. It is of interest as it may help humans to be aware of minute information about particular objects and reduce human tasks. Automatic recognition and extraction adds to the smart systems used today.

Moreover, the idea of making this project came as a part of a study carried out on the working and mechanism of “Automobiles of Future”. As we all know that, we will soon be seeing self-driving cars in future. In order for a car to decide what to do next: accelerate, apply brakes or turn, it needs to know where all the objects are around the car and what those objects are. That requires object detection. You would essentially train the car to detect known set of objects: cars, pedestrians, traffic lights, road signs, bicycles, motorcycles, etc.

2

### Objective:

The main problem of the people in India and even most of the countries is safety while driving. So, we came up with a solution to that by making use of this project to continuously scan the area for traffic, population etc. and as well as protection of the vehicles at the same time to prevent accidents or minor scratches to the vehicles.

Our main objective is to create an obstacle detector device which can be used for multiple purposes say it for car parking or security purposes.

After going through some of the papers regarding obstacle detection system implementation using open CV, we found that this concept is quite sought everywhere and is a popular concept which is still in progress. These papers had some innovative ideas for prevention from accidents and driving safer. The techniques that were illustrated were par excellence and can bring about a major change in the field of automobiles. The technologies used were not only efficient and reliable but also economically feasible. The existing system uses webcam and open CV and Visual studio C++, we have used Arduino UNO and processing software for respective purposes. Our major aim is tracking the object as accurately as possible.

3

# CHAPTER 2 LITERATURE SURVEY

### 2.1 Literature survey:

The Idea” Army, Navy and the Air Force make use of this technology. The use of such technology has been seen recently in the self-parking car systems launched by AUDI, FORD etc. And even the upcoming driverless cars by Google like Prius and Lexus. This setup can be used in any systems the customer may want to use like in a car, a bicycle or anything else. The use of Arduino in this provides even more flexibility of usage of the above-said module according to the requirements. The idea of making this project came as a part of a study carried out on the working and mechanism of “object tracking cameras”. Hence this time we were able to get a hold of one of the Arduino boards, Arduino UNO. So knowing about the power and vast processing capabilities of the Arduino, we thought of making it big and a day to day application specific module that can be used and configured easily at any place and by anyone.

Moreover, in this fast moving world there is an immense need for the tools that can be used for the betterment of the mankind rather than devastating their lives. Hence, from the idea of the self-driving cars came the idea of self-parking cars. The main problem of the people in the world is safety while driving. So, this gave up a solution to that by making use of this project to continuously scan the area for traffic, population etc. And as well as protection of the vehicles at the same time to prevent accidents or minor scratches to the vehicles.

# CHAPTER 3 ANALYSIS AND DESIGN

### Components used:

* + 1. **Arduino UNO:**

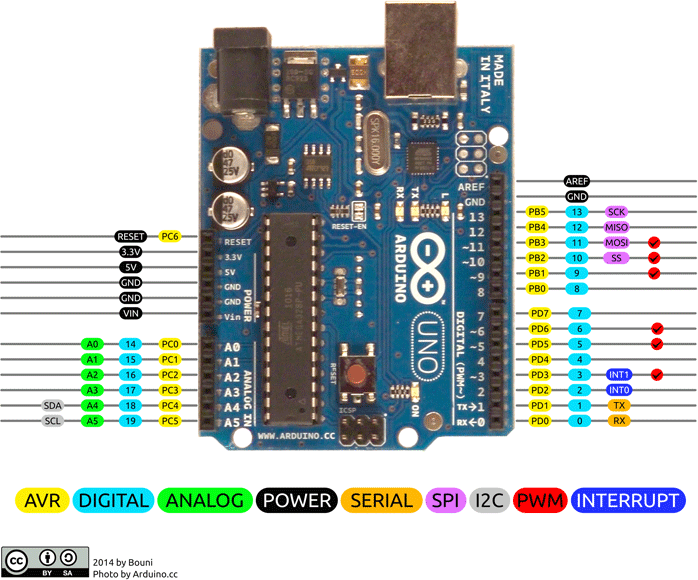


Fig 3.1

**Arduino Uno** is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button. Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The ATmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer.

* + - * The operating voltage is 5V
      * The input voltage ranges from 6v to 20V
      * Digital input/output pins are 14
      * Analog i/p pins are 6
      * DC Current for each input/output pin is 40 mA
      * DC Current for 3.3V Pin is 50 mA
      * Flash Memory is 32 KB
      * SRAM is 2 KB
    1. **Servo Motor SG-90:**

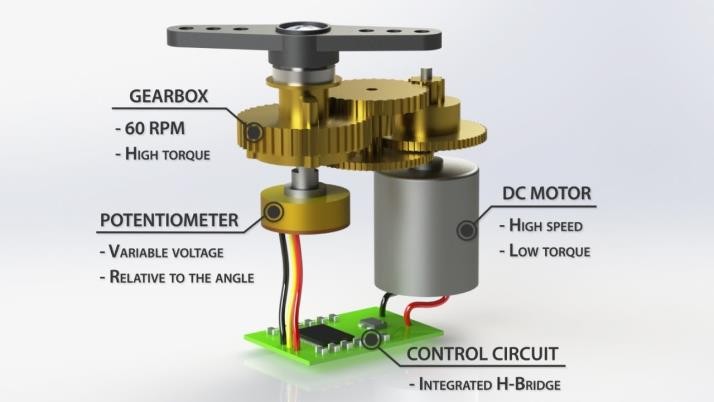


Fig 3.2

Servo motor works on the PWM (Pulse Width Modulation) principle, which means its angle of rotation is controlled by the duration of pulse applied to its control PIN. Basically, servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. As we know there are three wires coming out of this motor. To make this motor rotate, we must power the motor with +5V using the Red and Brown wire and send PWM signals to the Orange colour wire. Hence, we need something that could generate PWM signals to make this motor work, Arduino is used in our project for that purpose.

* + - * Operating Voltage is +5V typically
      * Torque: 2.5kg/cm
      * Operating speed is 0.1s/60°
      * Gear Type: Plastic
      * Rotation : 0°-180°
      * Weight of motor : 9gm

3.Breadboard

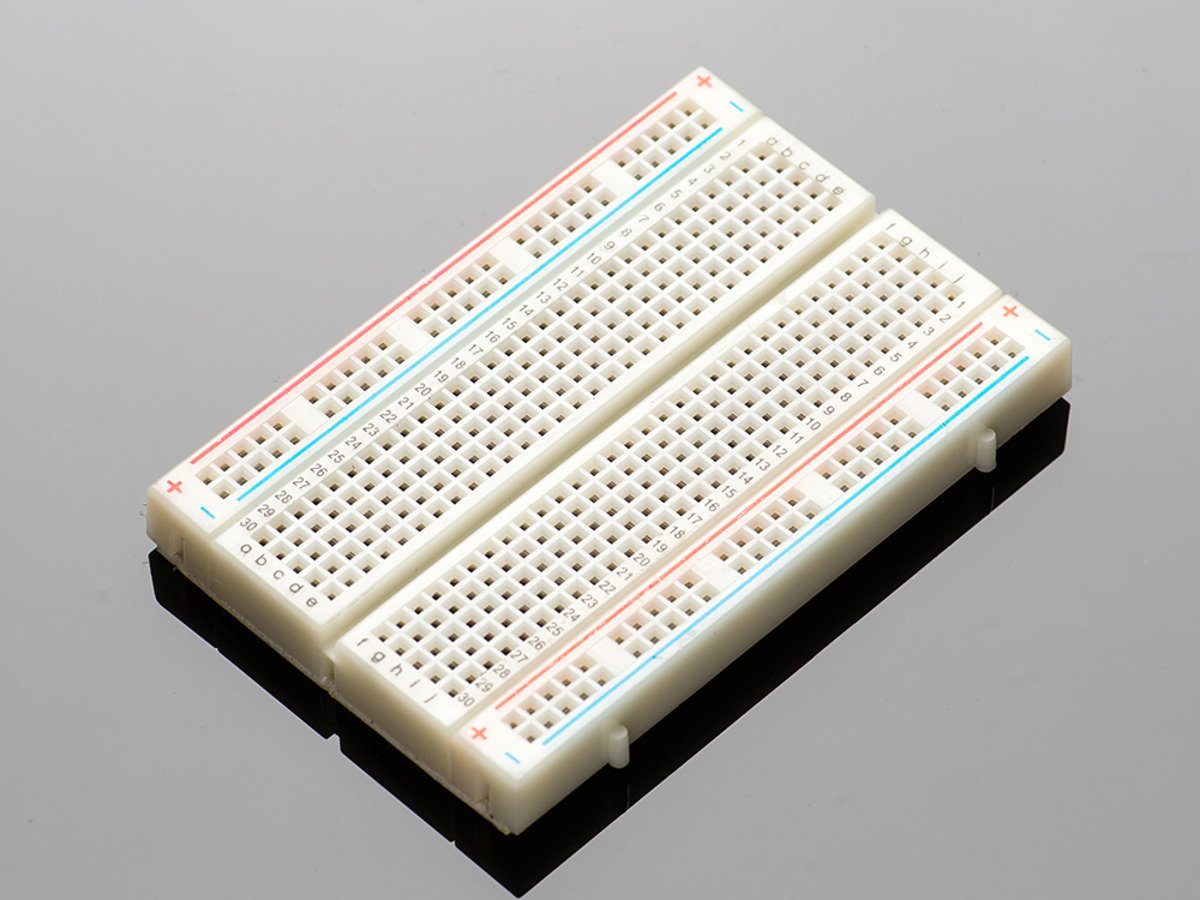


Fig 3.3

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.

10

4.Webcam

webcam gives input as a video frames to computer

>After receiving frames computer send them to Visual Studio C++ which contain open CV library running on it

>Then visual studio C++ program detects the image of the object from the webcam and calculate the coordinates of X,Y axis and radius of object

>from computer these coordinates or transferred to Arduino via serial communication between the Arduino and visual studio C++

>After receiving coordinates Arduino sends instructions to servomotors to move in X and Y directions

>camera can rotate 360 degrees to detect object around it



Fig 3.4

**11**

* + 1. **Arduino IDE:**

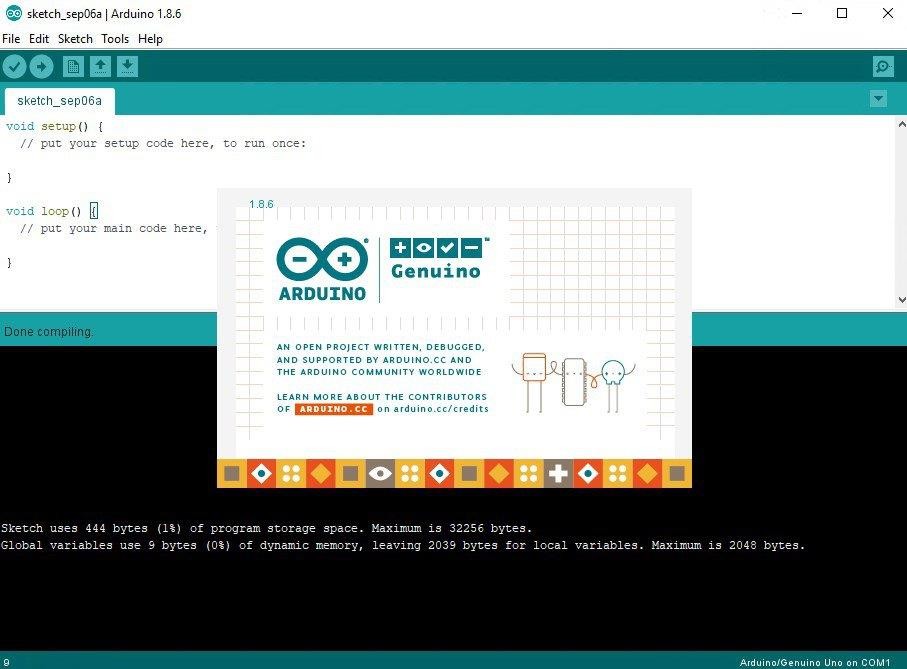


Fig 3.5

The Arduino Integrated Development Environment ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment)) is a [cross-](https://en.wikipedia.org/wiki/Cross-platform)

[platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS,](https://en.wikipedia.org/wiki/MacOS) [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++.](https://en.wikipedia.org/wiki/C%2B%2B_(programming_language))

It is used to write and upload programs to [Arduino](https://en.wikipedia.org/wiki/Arduino) compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the [GNU General Public](https://en.wikipedia.org/wiki/GNU_General_Public_License) [License,](https://en.wikipedia.org/wiki/GNU_General_Public_License) version 2.[[4]](https://en.wikipedia.org/wiki/Arduino_IDE#cite_note-4) The Arduino IDE supports the

languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring.

The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures.

User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program

stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU](https://en.wikipedia.org/wiki/GNU_toolchain) [toolchain,](https://en.wikipedia.org/wiki/GNU_toolchain) also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

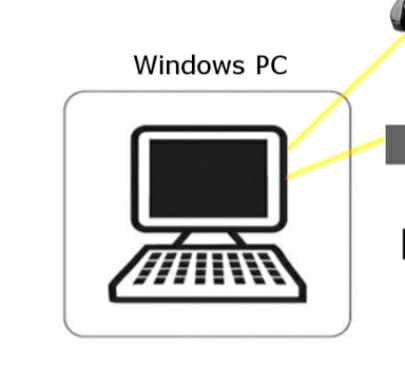
.

A picture containing text, electronics

Description automatically generated

Fig 3.6

Circuit diagram



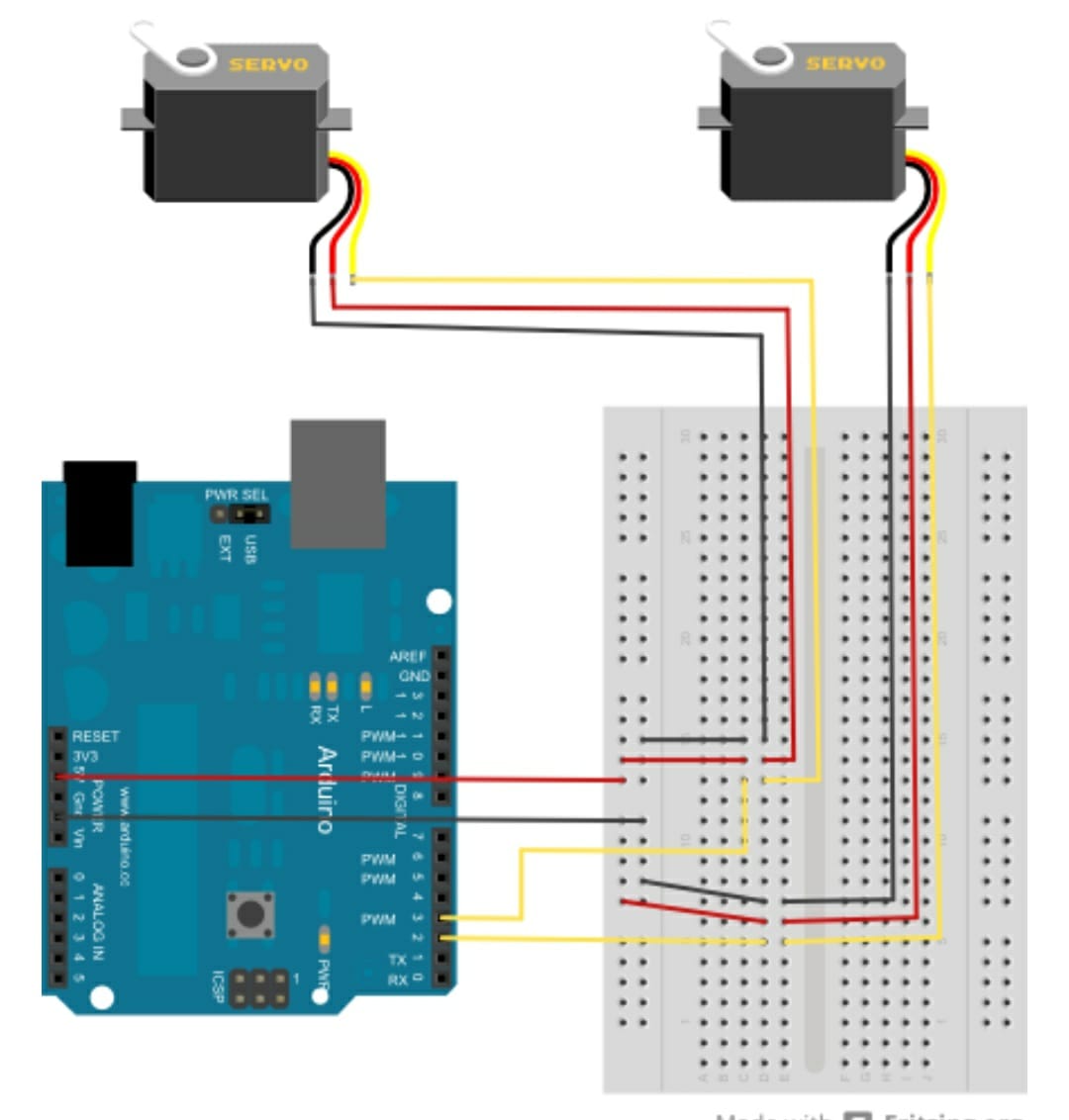


Fig 3.7

14

Working mechanism

* 1.Connect Lower Servo motor yellow wire with the PIN No - 9 of the arduino UNO. (controls x axis)
* 2.Connect Upper Servo motor yellow wire with the PIN No -10 of the arduino UNO(controls y axis)
* 3.Connect the red wire of the servo motor with 5v coming from arduino UNO
* 4.Connect the black wire of the servos withthe GND(Ground) coming from the arduinoUNO .

>> webcam gives input as a video frames to computer

>>After receiving frames computer send them to Visual Studio C++ which contain open CV library running on it

> >Then visual studio c++ program detects the image of the object from the webcam and calculate the coordinates of X,Y axis and radius of object

>>from computer these coordinates or transferred to Arduino via serial communication between the Arduino and visual studio C++

> >After receiving coordinates Arduino sends instructions to servomotors to move in X and Y directions

>>camera can rotate 360 degrees to detect object around it

15

# CHAPTER 4 IMPLEMENTATION

16

### Arduino source code:

#include <Servo.h>

int p\_fltXYRadius[0];

Servo servo;

Servo servo1;

int servoPosition = 90;

int servoPosition1=90 ;

int incomingByte = 0; // for incoming serial data

void setup()

{

Serial.begin(9600); // // opens serial port, sets data rate to 9600 bps

servo.attach(9); // attaches the servo on pin 9 to the servo object

servo1.attach(10);// attaches the servo1 on pin 10 to the servo object

servo.write(servoPosition); // set the servo at the mid position

servo.write(servoPosition1);// set the servo1 at the mid position

}

void loop()

{

if (Serial.available() > 0) {

// read the incoming byte:

incomingByte = Serial.read();

switch(incomingByte)

{

// Rotate camera left

case 'l':

servoPosition+=1;

delay(20);

if (servoPosition > 180)

{

servoPosition = 180;

}

break;

// Rotate camera right

case 'r':

servoPosition-=1;

delay(20);

if (servoPosition < 0)

{

servoPosition = 0;

}

break;

// Center camera

case 'c':

servoPosition = 90;

delay(20);

break;

// Camera in upward direction

case 'u':

servoPosition1+=5;

delay(100);

if (servoPosition1 > 160)

{

servoPosition1 = 160;

}

break;

// Camera in downward direction

case 'd':

servoPosition1-=5;

if (servoPosition1 < 140)

{

servoPosition1 = 140;

}

break;

// Camera would move upward if it finds an object moving up

case 'f':

servoPosition1+=1;

delay(100);

if (servoPosition1 > 180)

{

servoPosition1 = 180;

}

break;

// Camera would move downward if it finds an object moving down

case 'e':

servoPosition1-=1;

delay(100);

if (servoPosition1 < 0)

{

servoPosition1 = 0;

}

break;

}

servo.write(servoPosition);

servo1.write(servoPosition1);

}

}

### 4.2.visual studio c++ software source code:

#include <iostream>

#include<opencv/cvaux.h>

#include<opencv/highgui.h>

#include<opencv/cxcore.h>

#include <sstream>

#include <string>

#include <opencv\cv.h>

#include<stdio.h>

#include<stdlib.h>

// Need to include this for serial port communication

#include <Windows.h>

///////////////////////////////////////////////////////////////////////////////////////////////////

int H\_MIN = 0;

int H\_MAX = 256;

int S\_MIN = 0;

int S\_MAX = 256;

int V\_MIN = 0;

int V\_MAX = 256;

using namespace cv;

const string trackbarWindowName = "Trackbars";

void on\_trackbar( int, void\* )

{//This function gets called whenever a

// trackbar position is changed

}

void createTrackbars(){

//create window for trackbars

namedWindow(trackbarWindowName,0);

//create memory to store trackbar name on window

char TrackbarName[50];

sprintf( TrackbarName, "H\_MIN", H\_MIN);

sprintf( TrackbarName, "H\_MAX", H\_MAX);

sprintf( TrackbarName, "S\_MIN", S\_MIN);

sprintf( TrackbarName, "S\_MAX", S\_MAX);

sprintf( TrackbarName, "V\_MIN", V\_MIN);

sprintf( TrackbarName, "V\_MAX", V\_MAX);

//create trackbars and insert them into window

//3 parameters are: the address of the variable that is changing when the trackbar is moved(eg.H\_LOW),

//the max value the trackbar can move (eg. H\_HIGH),

//and the function that is called whenever the trackbar is moved(eg. on\_trackbar)

// ----> ----> ---->

createTrackbar( "H\_MIN", trackbarWindowName, &H\_MIN, H\_MAX, on\_trackbar );

createTrackbar( "H\_MAX", trackbarWindowName, &H\_MAX, H\_MAX, on\_trackbar );

createTrackbar( "S\_MIN", trackbarWindowName, &S\_MIN, S\_MAX, on\_trackbar );

createTrackbar( "S\_MAX", trackbarWindowName, &S\_MAX, S\_MAX, on\_trackbar );

createTrackbar( "V\_MIN", trackbarWindowName, &V\_MIN, V\_MAX, on\_trackbar );

createTrackbar( "V\_MAX", trackbarWindowName, &V\_MAX, V\_MAX, on\_trackbar );

}

///////////////////////////////////////////////////////////////////////////////////////////////////

int main(int argc, char\* argv[])

{

// Setup serial port connection and needed variables.

HANDLE hSerial = CreateFile(L"COM2", GENERIC\_READ | GENERIC\_WRITE, 0, 0, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

if (hSerial !=INVALID\_HANDLE\_VALUE)

{

printf("Port opened! \n");

DCB dcbSerialParams;

GetCommState(hSerial,&dcbSerialParams);

dcbSerialParams.BaudRate = CBR\_9600;

dcbSerialParams.ByteSize = 8;

dcbSerialParams.Parity = NOPARITY;

dcbSerialParams.StopBits = ONESTOPBIT;

SetCommState(hSerial, &dcbSerialParams);

}

else

{

if (GetLastError() == ERROR\_FILE\_NOT\_FOUND)

{

printf("Serial port doesn't exist! \n");

}

printf("Error while setting up serial port! \n");

}

char outputChars[] = "c";

DWORD btsIO;

// Setup OpenCV variables and structures

CvSize size640x480 = cvSize(640, 480); // use a 640 x 480 size for all windows, also make sure your webcam is set to 640x480 !!

CvCapture\* p\_capWebcam; // we will assign our web cam video stream to this later . . .

IplImage\* p\_imgOriginal; // pointer to an image structure, this will be the input image from webcam

IplImage\* p\_imgProcessed; // pointer to an image structure, this will be the processed image

IplImage\* p\_imgHSV; // pointer to an image structure, this will hold the image after the color has been changed from RGB to HSV

// IPL is short for Intel Image Processing Library, this is the structure used in OpenCV 1.x to work with images

CvMemStorage\* p\_strStorage; // necessary storage variable to pass into cvHoughCircles()

CvSeq\* p\_seqCircles; // pointer to an OpenCV sequence, will be returned by cvHough Circles() and will contain all circles

// call cvGetSeqElem(p\_seqCircles, i) will return a 3 element array of the ith circle (see next variable)

float\* p\_fltXYRadius; // pointer to a 3 element array of floats

// [0] => x position of detected object

// [1] => y position of detected object

// [2] => radius of detected object

int i; // loop counter

char charCheckForEscKey; // char for checking key press (Esc exits program)

p\_capWebcam = cvCaptureFromCAM(0); // 0 => use 1st webcam, may have to change to a different number if you have multiple cameras

if(p\_capWebcam == NULL) { // if capture was not successful . . .

printf("error: capture is NULL \n"); // error message to standard out . . .

getchar(); // getchar() to pause for user see message . . .

return(-1); // exit program

}

// declare 2 windows

cvNamedWindow("Original", CV\_WINDOW\_AUTOSIZE); // original image from webcam

cvNamedWindow("Processed", CV\_WINDOW\_AUTOSIZE); // the processed image we will use for detecting circles

createTrackbars();

p\_imgProcessed = cvCreateImage(size640x480, // 640 x 480 pixels (CvSize struct from earlier)

IPL\_DEPTH\_8U, // 8-bit color depth

1); // 1 channel (grayscale), if this was a color image, use 3

p\_imgHSV = cvCreateImage(size640x480, IPL\_DEPTH\_8U, 3);

// Variables for Arduino Control

int servoPosition = 90;

int servoOrientation = 0;

int servoPosition1=90;

int servoOrientation1=0;

// Main program loop

while(1) { // for each frame . . .

p\_imgOriginal = cvQueryFrame(p\_capWebcam); // get frame from webcam

if(p\_imgOriginal == NULL) { // if frame was not captured successfully . . .

printf("error: frame is NULL \n"); // error message to std out

getchar();

break;

}

// Change the color model from RGB (BGR) to HSV. This makes it easier to choose a color based on Hue

cvCvtColor(p\_imgOriginal, p\_imgHSV, CV\_BGR2HSV);

cvInRangeS(p\_imgHSV, // function input

cvScalar(H\_MIN, S\_MIN, V\_MIN), // min filtering value (if color is greater than or equal to this)

cvScalar(H\_MAX, S\_MAX, V\_MAX), // max filtering value (if color is less than this)

p\_imgProcessed); // function output

p\_strStorage = cvCreateMemStorage(0); // allocate necessary memory storage variable to pass into cvHoughCircles()

// smooth the processed image, this will make it easier for the next function to pick out the circles

cvSmooth(p\_imgProcessed, // function input

p\_imgProcessed, // function output

CV\_GAUSSIAN, // use Gaussian filter (average nearby pixels, with closest pixels weighted more)

9, // smoothing filter window width

9); // smoothing filter window height

// fill sequential structure with all circles in processed image

p\_seqCircles = cvHoughCircles(p\_imgProcessed, // input image, nothe that this has to be grayscale (no color)

p\_strStorage, // provide function with memory storage, makes function return a pointer to a CvSeq

CV\_HOUGH\_GRADIENT, // two-pass algorithm for detecting circles, this is the only choice available

2, // size of image / 2 = "accumulator resolution", i.e. accum = res = size of image / 2

p\_imgProcessed->height / 4, // min distance in pixels between the centers of the detected circles

100, // high threshold of Canny edge detector, called by cvHoughCircles

50, // low threshold of Canny edge detector, called by cvHoughCircles

10, //10 // min circle radius, in pixels

400); // max circle radius, in pixels

// Run this if the camera doesn't detect any circles

if (p\_seqCircles->total == 0)

{

// Initialize orientation

// This just makes it so the camera first goes to the side that it's leaning towards

// So if the camera is already mostly facing the left side it goes to the left end

// before going to the right. And the other way around.

if (servoOrientation == 0)

{

if (servoPosition >= 90)

servoOrientation = 1;

else

servoOrientation = -1;

}

if (servoOrientation == 1)

{

outputChars[0] = 'l';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

// This code is identical to the one on the Arduino side

servoPosition+=1;

if (servoPosition > 180)

{

servoPosition = 180;

servoOrientation = -1;

}

}

else

{

outputChars[0] = 'r';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

// This code is identical to the one on the Arduino side

servoPosition-=1;

if (servoPosition < 0)

{

servoPosition = 0;

servoOrientation = 1;

}

}

///////////////////////////////////////////////////////////////////////////////////

if (servoOrientation1 == 0)

{

if (servoPosition1 >= 150)

servoOrientation1 = 1;

else

servoOrientation1 = -1;

}

if (servoOrientation1 == 1)

{

outputChars[0] = 'u';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

// This code is identical to the one on the Arduino side

servoPosition1+=3;

if (servoPosition1 > 160)

{

servoPosition1 = 160;

servoOrientation1 = -1;

}

}

else

{

outputChars[0] = 'd';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

// This code is identical to the one on the Arduino side

servoPosition1-=3;

if (servoPosition1 < 140)

{

servoPosition1 = 140;

servoOrientation1 = 1;

}

}

///////////////////////////////////////////////////////////////////////////////////

}

// Run this if the camera can see at least one circle

//for(i=0; i < p\_seqCircles->total; i++) { // for each element in sequential circles structure (i.e. for each object detected)

if (p\_seqCircles->total == 1)

{

p\_fltXYRadius = (float\*)cvGetSeqElem(p\_seqCircles, 1); // from the sequential structure, read the ith value into a pointer to a float

printf("ball position x = %f, y = %f, r = %f \n", p\_fltXYRadius[0], // x position of center point of circle

p\_fltXYRadius[1], // y position of center point of circle

p\_fltXYRadius[2]); // radius of circle

// Reset servo orientation as the camer a now has focus of a circle

// Servo orientation is important only when the camera doesn't see a circle

servoOrientation = 0;

//char outputChars[] = "p\_fltXYRadius[0]";

//WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

//char outputChars1[] = "p\_fltXYRadius[1]";

//WriteFile(hSerial, outputChars1, strlen(outputChars1), &btsIO, NULL);

// Check whether camera should turn to its left if the circle gets near the right end of the screen

if (p\_fltXYRadius[0] > 400)

{

outputChars[0] = 'l';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

servoPosition+=1;

if (servoPosition > 180)

servoPosition = 180;

}

// Check whether camera should turn to its right if the circle gets near the left end of the screen

if (p\_fltXYRadius[0] < 240)

{

outputChars[0] = 'r';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

servoPosition-=1;

if (servoPosition < 0)

servoPosition = 0;

}

///////////////////////////////////////////////////////////////////

servoOrientation1 = 0;

if (p\_fltXYRadius[1] > 320)

{

outputChars[0] = 'e';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

servoPosition1+=1;

if (servoPosition1 > 180)

servoPosition1 = 180;

}

// Check whether camera should turn to its right if the circle gets near the left end of the screen

if (p\_fltXYRadius[1] < 160)

{

outputChars[0] = 'f';

WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

servoPosition1-=1;

if (servoPosition1 < 0)

servoPosition1 = 0;

}

///////////////////////////////////////////////////////////////////

// draw a small green circle at center of detected object

cvCircle(p\_imgOriginal, // draw on the original image

cvPoint(cvRound(p\_fltXYRadius[0]), cvRound(p\_fltXYRadius[1])), // center point of circle

3, // 3 pixel radius of circle

CV\_RGB(0,255,0), // draw pure green

CV\_FILLED); // thickness, fill in the circle

// draw a red circle around the detected object

cvCircle(p\_imgOriginal, // draw on the original image

cvPoint(cvRound(p\_fltXYRadius[0]), cvRound(p\_fltXYRadius[1])), // center point of circle

cvRound(p\_fltXYRadius[2]), // radius of circle in pixels

CV\_RGB(255,0,0), // draw pure red

3); // thickness of circle in pixels

} // end for

cvShowImage("Original", p\_imgOriginal); // original image with detectec ball overlay

cvShowImage("Processed", p\_imgProcessed); // image after processing

cvReleaseMemStorage(&p\_strStorage); // deallocate necessary storage variable to pass into cvHoughCircles

charCheckForEscKey = cvWaitKey(10); // delay (in ms), and get key press, if any

if(charCheckForEscKey == 27) break; // if Esc key (ASCII 27) was pressed, jump out of while loop

} // end while

cvReleaseCapture(&p\_capWebcam); // release memory as applicable

cvDestroyWindow("Original");

cvDestroyWindow("Processed");

// This closes the Serial Port

CloseHandle(hSerial);

return(0);

# CHAPTER 5 RESULTS

28

### 5.1 Simulation Results:

The following images are few images of our project after complete implementation.

A picture containing electronics, connector, adapter

Description automatically generated

Fig 5.1

The above image shows, how the connections are made, and how webcam is

attached on top of the servo motor. The servo motor rotates from 0 degrees to 180 degrees and if detects ant object, the object distance and angle is viewed on the lcd screen.

The same values can be viewed in the serial monitor of Arduino IDE.

29

If everything was done correctly then.   
  
Start the program and 4 screens would open up...  
  
1. First screen would be for selecting the appropriate HSV values so that u can opt for particular object you want to track for.

2.Second screen would show the original image of your webcam.

3.Third screen would show the processed image of your webcam usually a binary image.

4.Fourth screen would show u the x axis, y axis and radius of that image.

30

# CHAPTER 6

APPLICATIONS, ADVANTAGES AND LIMITATIONS

### Applications:

Object detection is breaking into a wide range of industries, with use cases ranging from personal security to productivity in the workplace. Object detection and recognition is applied in many areas of computer vision, including image retrieval, security, surveillance, automated vehicle systems and machine inspection. Significant challenges stay on the field of object recognition. The possibilities are endless when it comes to future use cases for object detection.

* Tracking objects
* Face detection
* Optical character recognization
* Traffic monitoring
* Robot vision
* Security and Survelliance

### Advantages:

#### Low cost of implementation:

We use the very basic materials like a microcontroller,aurdino. So, our cost of making the product is very less as compared to many other existing object detection systems. This gives an an opportunity of easy affordability and usage by end users. Also getting good results by implementing this system is our main aim with lowest price possible.

#### Readings are updated rapidly:

As our system is dynamic in nature camera are active all the time. So, when ever change spotted it detects the objects and information is sent to micro visual studio all the time. It helps in maintaining reliability.

#### Less power consumption:

As we know that we are in an era where "Power management" plays a main role in today's world. As we are getting shortage of power these days it's very important to use low power consumption devices. Compared to many other object detection systems which uses cameras and many other sensors, where a lot of power is required to process the huge information, our system is manages to do the task with fraction of that power.

32

#### Easy to design:

Designing of a product for an application is one of the major thing . Generally it needs a lot of time and effort. But in our "Object detection system" design part is a lot easier as it takes only a few days to do so . First we write software for the working as per our requirements then we assemble the hardware and link the physical and software parts with an IDE.

### Limitations:

Each and every system will have both advantages and limitations. But it is important to note that there are significantly more advantages and very limited number of limitations. By considering all the above-mentioned advantages and to meet the applications the following are the limitations posed by our system.

* + 1. Height of objects cannot be determined.
    2. 3D mapping of object is not possible.
    3. The range depends upon the characteristics of the camera used
    4. It would be a problem if the objects are moving with high speed

# CHAPTER 7 CONCLUSION AND FUTURE SCOPE

* 1. Conclusion:

Using this object detector parking becomes a lot easier. It also helps in avoiding collisions, also in theft prevention and lot more applications.

It is considerably less price so it can be widely used and implemented. It consumes less power and operates with good precision. Customer interaction is way more easier than with many other devices.

* 1. Future scope:

Our project would have a great impact on object detection in automotive, robotics and home automation industry. Specifically mentioning about the parking and collision detection, the data will be sent to the cloud and analysis is done so that work can be done more efficiently in future. This idea can also be included in driverless cars.

### REFERENCES

### >>[http://aleksandarkrstikj.com/tracking-a-ball-and-rotating-camera-with- opencv-and-arduino/](http://aleksandarkrstikj.com/tracking-a-ball-and-rotating-camera-with-%20%20%20%20%20%20opencv-and-arduino/)

>>[https://www.instructables.com/id/Face-detection-and-tracking-with-Arduino- and-OpenC/](https://www.instructables.com/id/Face-detection-and-tracking-with-Arduino-%20%20%20and-OpenC/)

>><https://www.sparkfun.com/tutorials/304>

>><https://www.instructables.com/OBJECT-DETECTION-AND-TRACKING-USING-OPENCV-VISUAL-/>

>>International journal of computer science and mobile computing

.