GUI based Arduino controlled Robotic Arm using Colour detection and Speech recognition

Project Report

Submitted by

Ananth S
120906758

Rahul Yadav
120906832

Under the guidance of

Dr. B. K. Singh
Professor
Manipal Institute of Technology, Manipal University

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MANIPAL INSTITUTE OF TECHNOLOGY
(A Constituent College of Manipal University)
MANIPAL – 576104, KARNATAKA, INDIA

May 2016

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent College of Manipal University)

MANIPAL – 576 104 (KARNATAKA STATE), INDIA

Manipal

Date

CERTIFICATE

This is to certify that the project titled **GUI based Arduino controlled Robotic Arm using Colour detection and Speech recognition** is a record of the bonafide work done by **Ananth S** (120906758) and **Rahul Yadav** (120906832) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.E) in Electrical and Electronics Engineering (E & E Engg) by Manipal Institute of Technology Manipal, Karnataka (A Constituent College of Manipal University), during the academic year 2015-16).

Dr. B. K. Singh	**Dr. Savitha G. Kini**
Professor, Dept of E & E Engg	*HOD, Dept of E & E Engg*

ACKNOWLEDGMENTS

Our deep gratitude goes to our project guide Dr. B. K. Singh (Professor Dept. of E & E) for guiding us during the project work. We would like to show our gratitude to our Dept. HOD Dr. Savitha G. Kini for motivating us to do the best of our ability for completion of this project. We would like to show our deepest gratitude to our Director of our esteemed college Dr. Gopalkrishna Prabhu for providing us the opportunity to work on this project in our last semester. We would like to thank all the teaching and non teaching staff who helped us in completion of the project.

ABSTRACT

The Demand for consumer goods and assembly line manufactured items have sky-rocketed in the recent days. Right from candies to cars, industries wish to produce ‘more’ in ‘less’ time. Industries have thus witnessed the automation of its processes, increasing the efficiency and decreasing time, human labour and costs.

Depending on the type of application we can make robots to function much more effectively. One such basic application is in the packaging industry where it can be used for picking and placing the objects in their boxes. They can also perform laborious processes like spot welding in manufacturing industries if they are mounted with the welding apparatus and the gestures are programmed accordingly.

The structure of the arm was fabricated using a acrylic. The required shapes were drawn on a CAD software and were cut using a laser cutting machine. The signals in the form of either colour or voice are received as input to the MATLAB program through the camera or microphone of the computer. Further processing of this information resulted in the detection of the colour or recognition of the voice command. The MATLAB then sends the control signal to the Arduino. The Arduino drives the motors that cause the robotic arm to move.

The Project was completed successfully and the results were verifiable. The joints of the arm responded to different angles. The Programmed pre-sets worked efficiently. The voice recognition system was fair and could identify commands from a single user in noise-less environments. The colour detection system performed efficiently and recognized the RGB colours with precision.

On the whole, though the project cannot hit the industry immediately, it hold immense potential as voice controlled Robotic Arms are not very popular at this time. It was a marvellous experience and involved lots of learning!

LIST OF TABLES

LIST OF FIGURES

Contents

CHAPTER 1
INTRODUCTION

Robotics by nature is an inter-interdisciplinary field utilizing Digital Signal Processing Control Systems and Embedded Systems for their control, sensory feedback, and information processing. As Electrical Engineers we were inspired to pursue a project on these lines after reading a few papers on robotics and watching a TEDX video. The Arduino controlled robotic arm as we call it makes use of a camera or a microphone to either capture the colour codes or the voice commands made by the user. These signals are then processed by Matlab and the servo motors rotate thus causing the movement of the robotic arm.

1.1. Background and Motivation

Our motivation to pursue this project stems from the various research papers we had gone through and as we had stated above, a TEDX talk on robotics by Dr. Vijay Kumar, the dean of the University of Pennsylvania's School of Engineering and Applied Science. He had talked about how robotics was still in its nascent stages and suggested that the next 10 years would see wide scale research, development and deployment of robots for various purposes.

So how important is our work in today's world? Well to be honest this project is by no means a milestone in robotics, but having said that, it holds immense potential in fields of defence and automation industries. We have tried to leverage our knowledge of embedded systems, and MATLAB to build this robotic arm that can controlled by either colour codes, voice or by simply controlling each joint through the GUI

We were handicapped because of our some-what amateurish coding skills. Most of the projects we used as reference were coded on either C# or C++ on visual studio. But it this limitation that led us to explore other ways of implementing such a project. Thankfully Matlab's rather exhaustive documentation had inbuilt support packages for both image processing as well as Arduino. Our experience in the system simulation lab helped us overcome some of these challenges.

We aim to implement this project as a pick and drop robot. But as we had mentioned it holds great potential. With further research and development it may be possible to implement features such a packing in industries and automating assembly processes in industries.

1.2 Literature Review

Literature Survey:

1) D.Senthamaraikannan, S.Shriram, Dr. J. William "REAL TIME COLOR RECOGNITION", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control engineering (ACDT), 2015.[1]

Extracting primary colors for the purpose of vision-based human–computer interaction. Vision-based human–computer interaction could be achieved by analyzing segmented primary color regions. This paper presented color-based image segmentation, non-stationary color-based

target tracking, color based mouse pointer, color based virtual music instruments, and color based virtual calculator. Our experiments show that Red, green and blue are the default color used for recognition process. Live video was captured from camera and the video was converted into number of frame images. This algorithm monitored and process the every frame from the live video. Microphone and camera is used as an input device. Here image processing is done through MATLAB for color recognition process. We have implemented the same in our project as well.

2) Amab Pramanik, Rajorshee Raha, Automatic Speech Recognition using correlation analysis, World Congress on Information and Communication Technologies (WICT), 2012. [2]

Any speech recognition system feature extraction and patter matching are two very significant terms. In this paper we understood a simple algorithm for matching the patterns to recognize speech. We read about Mel frequency cepstral coefficients (MFCCs) as the feature of the recorded speech. This algorithm was implemented simply by using the principle of correlation. All the simulation experiments were carried out using MATLAB where the method produced relatively good results. This paper gave a details introduction of recorded speech processing, design considerations and evaluation results. We implemented this algorithm in our project too.

3) Krutarth Gandhi, Harsh Kotak, Susmit Joshi & Vikas Pandita, Motion Controlled Robotic arm, International Journal of Electronics and Communication Engineering (IJECE),2013. [3]

Controlling the robotic arm using potentiometers and turn-angle correlation between the potentiometer and servo motor as a result of a modified pulse width modulated signal generated from the Arduino.

4) Rajesh Kannan Megalingam, Nihil Saboo, Nitin Ajithkumar, SreeramUnny, Deepansh Menon, “Kinect Based Gesture Controlled Robotic Arm: A Research work at HuT Labs”, IEEE, 2013. [4]

Kinect senses the object using the combination of cameras and returns the coordinates in three dimension and the angles generated by the movement of the arm.

The proper functioning of the robotic based gesture controlled robotic arm necessitates the proper computation of angles formed by the various links and joints. From a mechanical point of view, there exist a number of joints and links spread thought the robotic arm but considering only the various tasks to be performed by the arm only three of them are to be monitored in real time .The angles aforementioned include both two dimensional and three dimensional

ones, which can only be determined by forward kinematics. The method of forward kinematics, seen in a very frequent basis in almost all the mechanical systems that have been designed up to date, involves calculating the position and orientation of the end effector of the robotic arm from the joint parameters. Serial communication is used to transfer the obtained angles to an Arduino micro controller, which then moves the robotic arm in requisite fashion.

1.3 Objectives

Main objective of this project is to build a robotic arm that can be controlled using voice commands and colour commands

In the process several secondary objectives would be accomplished which include:

1. Designing and Construction on a simple robotic arm using proprietary CAD Software.
2. Building a Graphical User Interface that moves each joint of the arm separately

1.4 Organization of the project report

Background Theory: This chapter deals with the algorithms used in this project.
Methodology: This chapter deals with the implementation of the software and the hardware used for the completion of the project.
Result Analysis: This chapter lists the result obtained for this project.
Conclusion and Future Scope: This chapter concludes the report by stating the strength and weakness of this project. It also deals with the future implementation of the project.

CHAPTER 2
BACKGROUND THEORY

In this chapter a look at the theory and principles behind the working of the project would be done with respect to the both image processing and voice recognition.

The project relies on two concepts for its working:
1. Image Processing
2. Simple Voice Recognition
We shall look in detail into each one of them now.

2.1 Image Processing:

So what exactly is Image processing?
Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.
It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography.
- Analysing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

Purpose of Image processing
The purpose of image processing is divided into 5 groups. They are:
- Visualization - Observe the objects that are not visible.
- Image sharpening and restoration - To create a better image.
- Image retrieval - Seek for the image of interest.
- Measurement of pattern – Measures various objects in an image.
- Image Recognition – Distinguish the objects in an image.

Here we use image processing to detect the presence of Red, Blue and Green coloured objects in the video stream. [1]

The principle here is "A Red object can be detected in an image by accessing the red channel of the image. Its location indicated by converting it into a binary image with the red coloured objects as white and masking the white region with a red coloured blob".

2.2 Speech Recognition

What is speech recognition? Where do I find it?

Speech recognition is the ability of a machine or program to identify words and phrases in spoken language and convert them to a machine-readable format.

Speech-recognition technology is embedded in voice-activated routing systems at customer call centres, voice dialling on mobile phones, and many other everyday applications. A robust speech-recognition system combines accuracy of identification with the ability to filter out noise and adapt to other acoustic conditions, such as the speaker's speech rate and accent. Designing a robust speech-recognition algorithm is a complex task requiring detailed knowledge of signal processing and statistical modelling.

The method employed here is very simple "The similarity between signals is done by the cross correlation function which returns plots of varying amplitude. The plot with the maximum amplitude corresponds to the matched signal" [2]

Now based on the voice command the Robotic Arm performs the required action.

Thus, in this chapter, the two guiding concept behind the project and their implementation was looked at. In the next chapter the entire hardware and software implementation has been explained in detail.

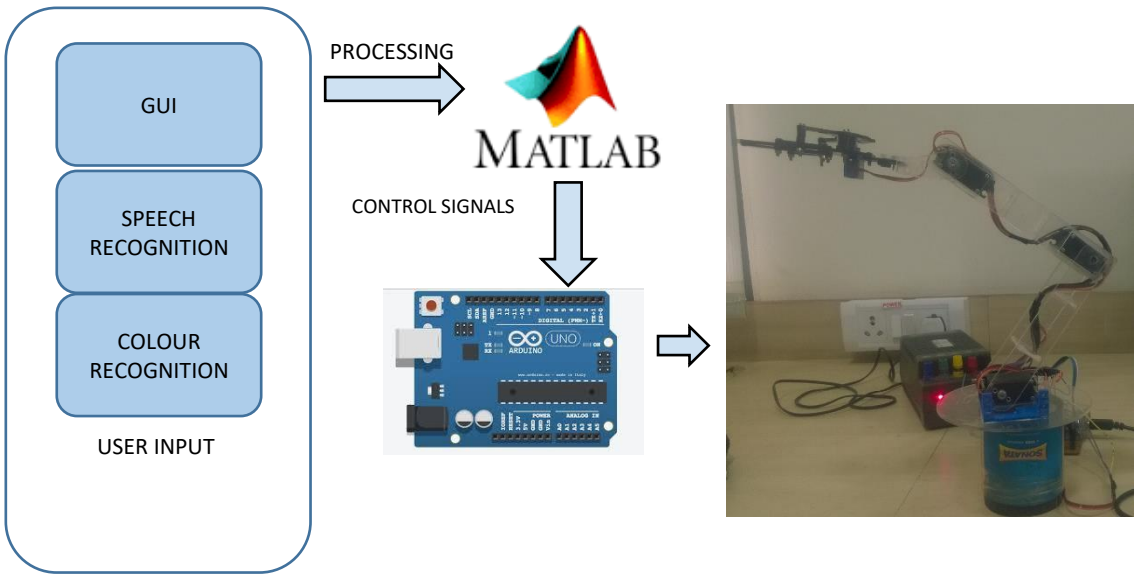


Figure 2.1 Basic Block Diagram

CHAPTER 3
METHODOLOGY

3.1 Introduction

In this chapter we shall be discussing in detail the proposed solution to our project by looking at the top-level block diagram identifying the input and outputs, the software used and a detailed explanation of the code. The hardware specifications too will be discussed with the help of relevant block diagrams and circuits and calculations.

3.2 Proposed Solution

The Robotic arm built can receive inputs either in the form of voice commands or colour codes. Besides, a Graphical user interface to control each of the servo motors along with options for pre-set gestures has been implemented. While the first program can identify red, blue and green coloured objects in the background, provided they meet the size and shape specifications, the second program is capable of recognizing voice commands. The output is in the form of a gesture performed by the robotic arm.

What really distinguishes our project from already existing ones is its ability to respond to voice, colour and manual control. It is thus a project that integrates visual, aural and touch based control. When we initially started we had planned to implement only colour recognizing aspect, but then we later added the additional features to make it a more wholesome project.

The main hardware components are:

1. The Arduino Uno microcontroller
2. The Servo Motors
3. The Arm Assembly
4. The camera and microphone of the laptop.

The main software components are

1. Matlab
2. The Arduino software package

Some of the assumptions we have made with respect to how the input is to be given are as follows:

- The colour codes or the objects that are used as inputs must meet the minimum size and shape specifications. This is done to prevent incorrect detection and haphazard motion of the arm.
- The voice command recognition is done by comparing the input voice signals with pre-recorded ones. Since each voice is characterised by a particular amplitude, frequency and accent it is assumed that the person who commands the robotic arm is the one who had pre-recorded the instructions.

The Robotic arm assembly was made in-house at the Robotics Lab at the innovation centre. The motors were selected based on the torque calculations. Thus the arm can handle only a limited amount of weight.

3.3 Implementation

3.3.1 Hardware Implementation

The construction and control of robotic arm requires different components. Five servo motors are used to control the position of the joints of the robotic arm. Arduino Uno microcontroller board is used to send the send the PWM control signal to each servo motor. Acrylic glass is used to make the robotic arm structure. The base for one of the motor was put in place using a 3D printed structure. Torque analysis of the robotic arm is done to choose the properly rated servo motors.

Arduino

Arduino Uno is a microcontroller board for rapid development and prototyping. It is based on the microcontroller by Atmel ATmega328P.

Table 3.1 Arduino Technical Specification

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Arduino Uno is used to provide the PWM signals to control the position of the servo motors which effectively moves the robotic arm in the desired way.
Following digital pins were used to control the servo motors.

- D5 – Shoulder (horizontal)
- D6 – Shoulder (vertical)
- D9 – Elbow
- D10 – Wrist
- D11 – Claw (Gripper actuator)

The PWM signal provided by the Arduino is 5V in amplitude and 50Hz frequency.

Servo Motors

Servo motors are specialised DC motor that contain utilises servomechanism for precise movement. It is a closed loop motor which senses position and uses it as feedback to control the motion and position of the motor. The control input signal specifies the current position of the motor output shaft and can be analogue or digital signal.
A typical servo motor contains a DC motor, a gearbox, an encoder and a controller.

- DC motor is the primary driving element of the servo motor.
- The gearbox is utilised to limit the speed, the extent of angular movement and to provide a high amount of stall torque.
- Encoder is used to sense speed and position of the shaft. A simple servo motor may contain only a position sensor which is achieved by using a potentiometer. Sophisticated and complex servo motors may use an optical rotary encoder to measure both speed of shaft and the current position.
- The control mechanism used in the servo motor is the bang-bang control or hysteresis control. In this type of control mechanism the motor is either on or off depending upon the error signal that comes from the controller. [3]

Table 3.2 Technical Specifications Of Servo Motor

Servo Motor	Tower Pro MG 995
Operating Voltage	4.8V – 7.2V
Modulation	Digital
Torque	9.4kg-cm (4.8V) – 13.5kg-cm (7.2V)
Rotation Angle	0º to 180º
Starting Position	90º
Maximum Current Rating	500mA
Weight	55g

A typical servo motor contains three wires

- Signal: A PWM signal is sent to control the position of the motor shaft.
- Supply: A DC power supply. Drives the motor.
- Ground: Ground wire for both the power supply and the control signal.

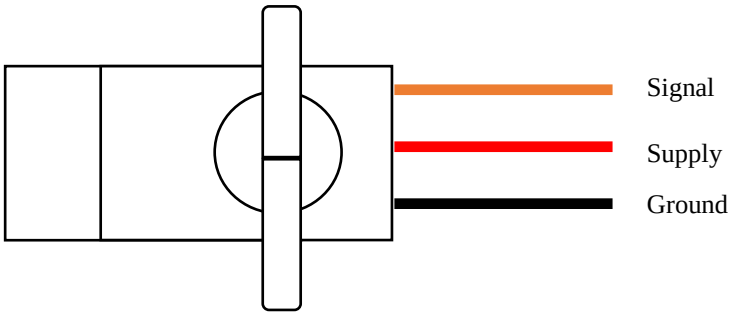


Figure 3.1 Servo Motor Wiring

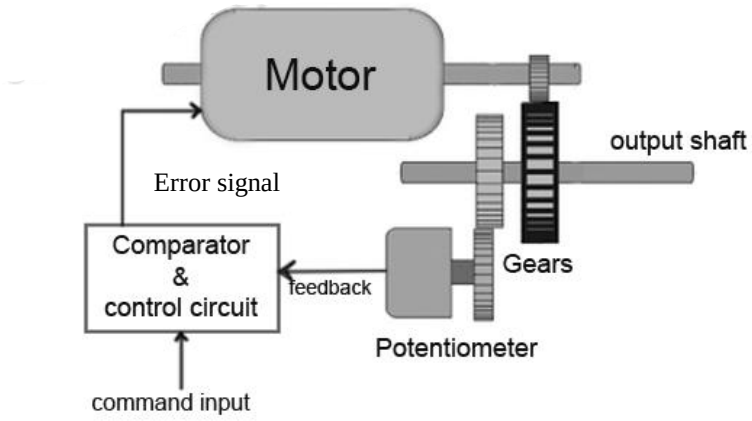


Figure 3.2 Servo Motor Block Diagram
http://www.electricaleasy.com/2015/01/how-does-servo-motor-work.html

As we can see from the figure when there is a command input signal it is first compared with the present position of the servo motor shaft which is determined by the potentiometer. So potentiometer acts as the feedback in this system. When the command input and the feedback from the potentiometer is same then the error from the comparator is zero, therefore there will be no movement of the shaft. When the error signal is non zero then the shaft rotates in one direction if the error signal is positive and in the opposite direction if the error signal is negative. The controller is proportional controller, which means the motor speed is proportional to the error signal. If the shaft's angular position is near the desired position (error signal is small) then the speed will be slower. If the shaft's angular position is farther away from the desired position (error signal is large) then the speed will be higher.

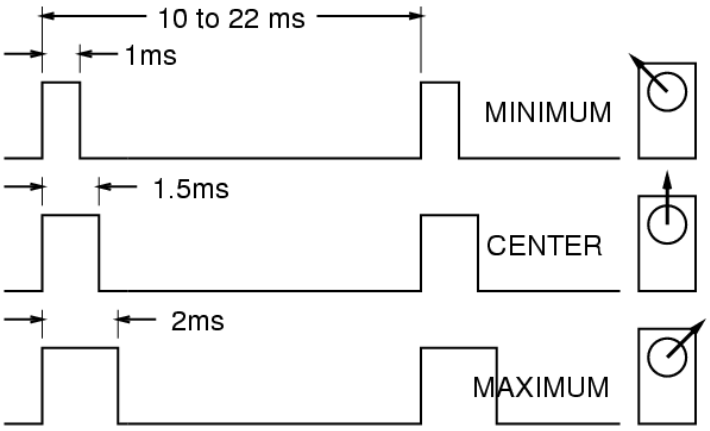


Figure 3.3 Servo Motor Position Using Pulses

http://projectedneuralactivity.blogspot.in/2012/12/controlling-servo-using-raspberry-pi.html

The input signal is pulse width modulated (PWM) signal that can be sent by a motor driver or a microcontroller.
In this case Arduino Uno microcontroller board is used to send the PWM signals to the servo motors.
When a pulse train of 1ms is supplied to the servo motor it rotates to ‘zero degree’ position or minimum position.
When the pulse train is of 2ms the servo motor rotates it rotates to ‘180 degree’ position or maximum position.
When the pulse train is of 1.5ms the servo motor goes to the default position or ’90 degree’ position.

Torque Analysis

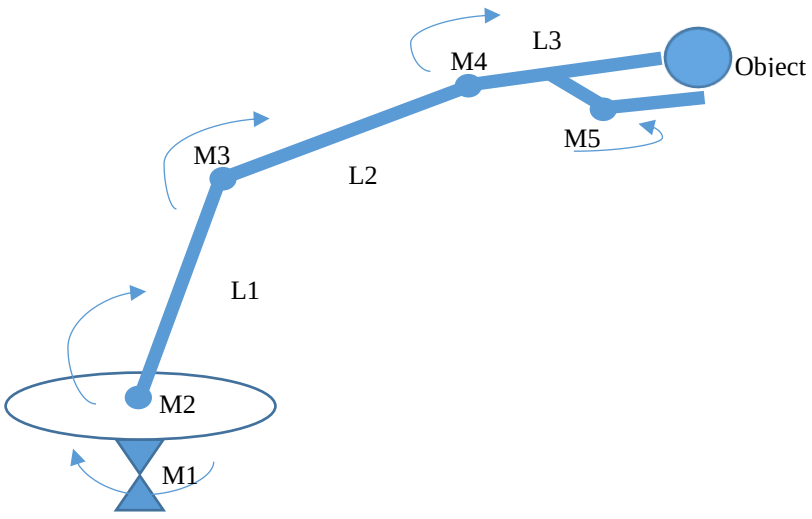


Figure 3.4 Robotic Arm Representation

Motors

- M1 – Shoulder (horizontal motion)
- M2 – Shoulder (vertical motion)
- M3 – Elbow
- M4 – Wrist
- M5 – Claw (Gripper actuator)

Links

- L1 – Distance between M2 and M3 – 18.5 cm
- L2 – Distance between M3 and M4 – 17.4 cm
- L3 – Distance between M4 and object - 16 cm

Masses

- ML1 – 40 g
- ML2 – 30 g
- ML3 – 155g
- M_{Object} – 50g
- Motors M1-M5 – 55g

Above is a pictorial representation of the robotic arm with all the motor position along with the length and weight of each links and motors.
The torque is calculated at each motor position i.e. motors M1 to M4. Motor M5 drives a gripper and has a rating of 1.6kg-cm.
To calculate the torque required we should assume the scenario where the torque required is more. When any of the link is parallel to the ground the distance between the joint and the end of the link is maximum.

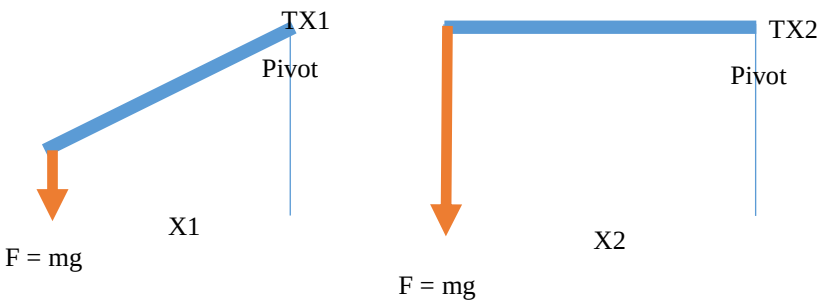


Figure 3.5 Maximum Torque Calculation

We can see as the link is rotated at the pivot point the distance increases from X1 to X2. Thus we can say that the torque TX2 will be higher than the torque TX1.

The torque T1 and T2 are same because they are present at the same joint. Only difference is that they rotate in different planes.

$$T2 = (L1 + L2 + L3) * \text{MObject} + \left(\text{L1} + \text{L2} + \frac{\text{L3}}{2}\right)\text{ML3} + (\text{L1} + \text{L2}) * \text{M4} + \left(\text{L1} + \frac{\text{L2}}{2}\right) * \text{ML2} + \text{L1} * \text{M3} + \frac{\text{L1}}{2} * \text{ML1} = 13.39 \text{ kgcm}$$

$$T3 = (L2 + L3) * \text{MObject} + \left(\text{L2} + \frac{\text{L3}}{2}\right)\text{ML3} + \text{L2} * \text{M4} + \left(\frac{\text{L2}}{2}\right) * \text{ML2} = 6.825 \text{ kgcm}$$

$$T4 = (L3) * \text{MObject} + \left(\frac{\text{L3}}{2}\right)\text{ML3} = 2.04 \text{ kgcm}$$

$$\text{T5} = 1.6 \text{ kgcm}$$

Table 3.3 Theoretical Value Of Torque Required At Each Joints

T1 at M1	13.39 kg-cm
T2 at M2	13.39 kg-cm
T3 at M3	6.825 kg-cm
T4 at M4	2.04 kg-cm
T5 at M5	1.6 kg-cm

The servo motor used TowerPro MG995 has a highest torque rating of 13.5 kg-cm when operated at 7.2V.
Using TowerPro MG995 servo motor at each joint will keep the operation of each servo motor under limit.
M_{object} should not exceed 100g for safe operation of the motors.

Acrylic Sheet

Poly(methyl methacrylate) or PMMA or simply acrylic glass is a lightweight transparent thermoplastic which is also shatter resistant. Acrylic has a density of 1.19 g/cm^3.
An acrylic sheet of 3mm was used to make the arm structure. A 2D model of the basic arm component was made which was later cut by a laser cutting machine. Each cut component was then glued together using a resin based glue. Servo motors were placed in these components using nuts and bolts.

3D Printing

To stick the base motor to rest of the assembly a 3D printed module was made which held the entire arm in place and gave support to the motor at the base.

3.3.2 Software Implementation

Matlab

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

1. Math and computation
2. Algorithm development
3. Modeling, simulation, and prototyping
4. Data analysis, exploration, and visualization
5. Scientific and engineering graphics
6. Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran.
MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

The MATLAB System

The MATLAB system consists of five main parts:

- The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

- The MATLAB working environment:

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

- Handle Graphics:

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

- The MATLAB mathematical function library:

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

- The MATLAB Application Program Interface (API):

This is a library that allows us to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

To acquire the video from the laptop's camera and the voice signal from the microphone we require a few MATLAB support packages as below:

Arduino package for Matlab:

Arduino programming is supposed to be fun but can become frustrating and time consuming for tasks such as plotting sensor data or incorporating advanced math, signal processing, or controls routines into your projects.
MATLAB and Simulink address several challenges with traditional Arduino programming.
The products support two primary workflows:

- Read, write, and analyze data from Arduino sensors http://in.mathworks.com/discovery/arduino-programming-matlab-simulink.html
- Develop algorithms that run standalone on the Arduino device

MATLAB support package for Arduino lets us write MATLAB programs that read and write data to the Arduino .Because MATLAB is a high level interpreted language, programming with it is easier than with C/C++ and other compiled languages, and results can be seen from I/O instructions immediately. MATLAB includes thousands of built-in math, engineering, and plotting functions that you can use to quickly to analyse and visualize data collected from your Arduino.

With MATLAB support package for Arduino, the Arduino is connected to a computer running MATLAB. Processing is done on the computer with MATLAB.
Benefits of using MATLAB for Arduino programming:

- Read and write sensor data interactively without waiting for the code to compile
- Analyse sensor data using thousands of pre-built functions for signal processing, machine learning, mathematical modelling
- Quickly visualize data using MATLAB's vast array of plot typeshttp://in.mathworks.com/products/matlab/plot-gallery.html http://in.mathworks.com/products/matlab/plot-gallery.html

OS Generic Video Interface Support Package:
We can use this support to bring live images from any webcam into MATLAB. This includes Webcams that may be built into laptops or other devices, as well as Webcams that plug into your computer via a USB port.

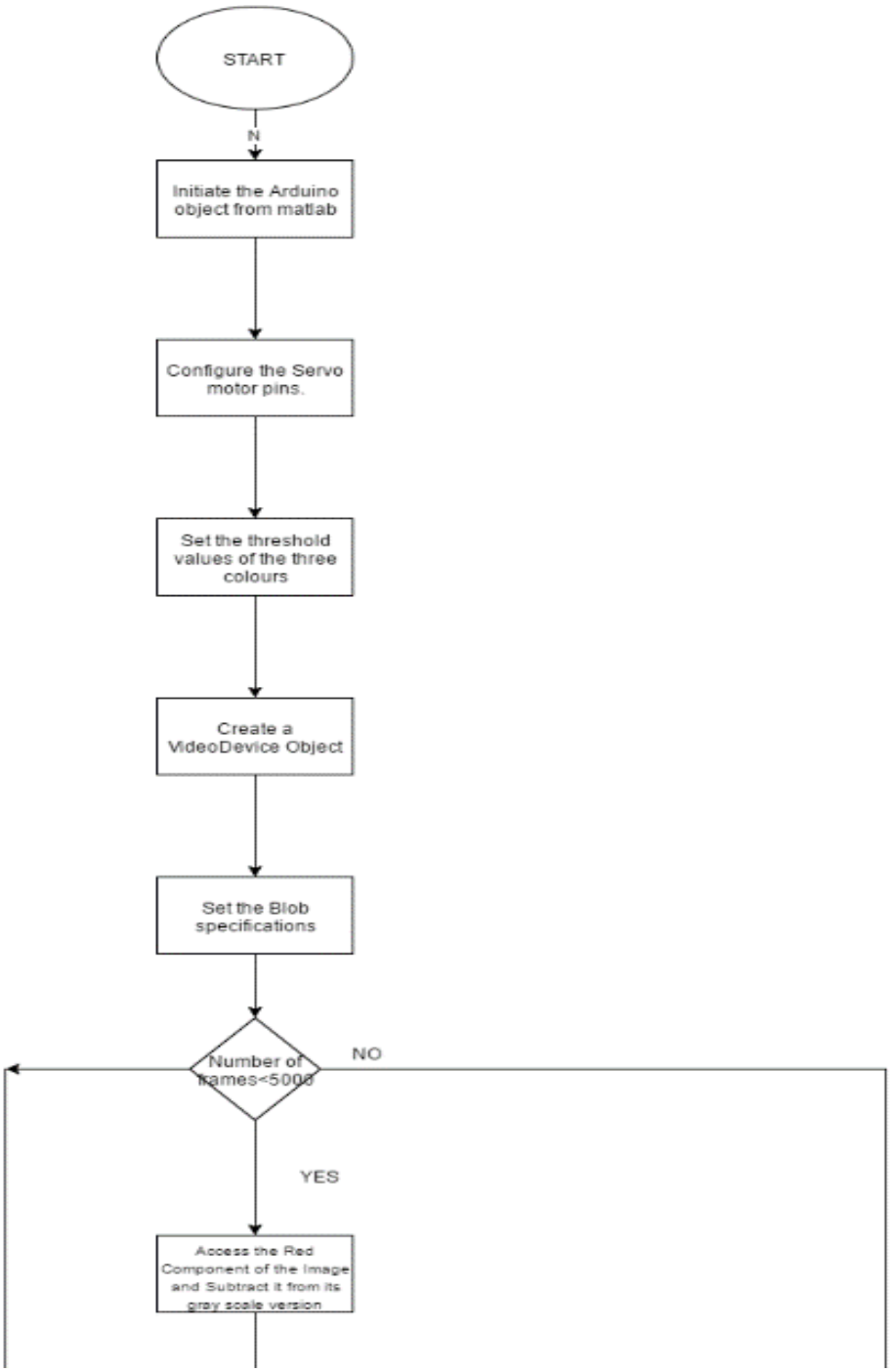
With this the set-up has been configured and ready to for programming.
As stated earlier we had worked on 3 independent methods to operate our Robotic Arm. We shall look into each one of them in detail:

Control using Colour Detection:

The red, blue and green coloured strips are recognized and the corresponding gesture is performed by the robotic arm.

The processing is done taking advantage of the image acquisition tool box which provides a complete environment for developing customized imaging solutions. It is possible to acquire images and video, visualize data, develop processing algorithms and analysis techniques, and create user interfaces (UIs) and apps.

The flowchart of the following program is:



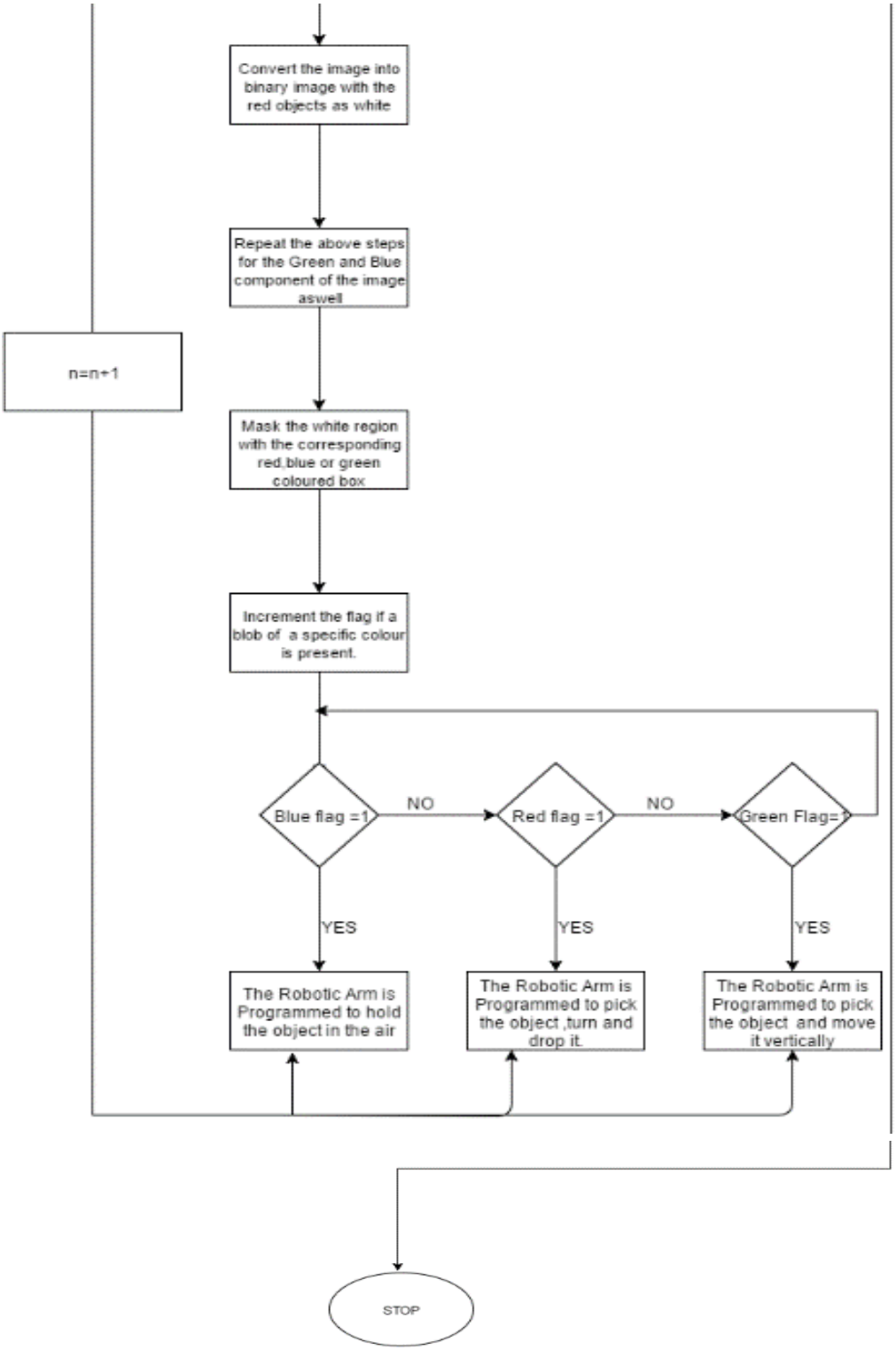


Figure 3.6 Colour Detection Flowchart

As Seen in the flow chart above the Arduino object is first initialized in-order to create a communication channel between MATLAB and the Arduino board. The required Arduino support package for MATLAB has to be installed for this to happen.

The threshold values for the detection of red blue and green are set. These values are ascertained by trial and error method. To acquire the video input an object of the Video Device. The command obj = imaq.VideoDevice (adaptorname, deviceid, format, ROI, Returned Colour space) creates a VideoDevice System object, where format is a text string that specifies a particular video format supported by the device or a device configuration file (also known as a camera file).The ROI parameter specifies the Region-of-interest for acquisition. This is set to the default ROI value for the specified device, which is the maximum resolution possible for the specified format. The format is 1-based, that is, it is specified in pixels in a 1-by-4 element vector [x y width height].The Returned colour space specifies the colour space of the returned image. The default value of the property depends on the device and the video format selected.

Next, the blob analysis parameters which include the Area Output Port, Centroid Output Port, Minimum Blob area, Maximum Blob Area and Maximum Count are specified. Shapes too are configured to be used at a later stage.

The command to start the Video stream is written. About 5000 frames are acquired and processed. In-order to obtain the presence of a red coloured object. The red component of the image is obtained. The noise is filtered out using a median filter because of its advantages while using with images and finally the image is converted into a binary image which can be expressed as a matrix. This is repeated for the green and blue colours as well

Now that the presence of a red, blue or green coloured object has been identified, the shapes that were declared above are coloured with the respective colours and inserted at the position where the coloured object is found. We are basically masking the white coloured zone indicative of the red, blue or the green coloured object.

Three counter's br, bg and bb are declared and their statuses are updated in real time depending upon the presence of red, green or blue coloured objects. The constraints for the blob are given in such a way that, at any time only one blob is recognized. Hence for example if br is 1 then the arm picks the object rotates and drops it down. Similarly for green it picks an object and moves it vertically and for blue it holds it in the air.

This process continues till the number of frames is 5000.After this, the memory and buffers are released.

Control of the Robotic Arm by speech

The Robotic Arm is programmed to react to voice commands as well. Commands such as "Pick the object", "Hold up" and "Drop it Down" are used to perform the gestures as indicated by them.

Now what is the algorithm employed to perform this action?

As shown in the flow chart, the Arduino object is initialized to create a channel of communication between Matlab and the Arduino. Three sound files each consisting of a command are recorded in a noise-less environment.

An audio recorder object is created to record the command to be given by the user. The command recordblocking records the sound for the give duration of time, specified as a parameter. The recorded sound is then saved in the hard disk using the audiowrite command.

The xcorr function returns the cross-correlation of two discrete-time sequences, x and y. Cross-correlation measures the similarity between x and shifted (lagged) copies of y as a function of the lag. If x and y have different lengths, the function appends zeros at the end of the shorter vector so it has the same length, *N*, as the other.

Using this xcorr function each of the three pre-recorded sound files are compare with the user command. A plot depicting the result of the cross correlation function is then displayed. The absolute peak of each of the plot is determined and stored in the variables m1, m2 and m3. The variable m is the assigned to the maximum value among m1, m2 and m3.An If- Else ladder then is executed to determine the sound file with maximum similarity.

The function writePosition(s,position) writes the specified value to the specified servo on the Arduino hardware. Postion can vary from 0 to 1 with 0 corresponding to 0 angle and 1 corresponding to 180 degree angle. Accordingly the Robotic Arm will either pick the object and move it vertically or pick the object rotate horizontally and drop it down or hold it up in the air.

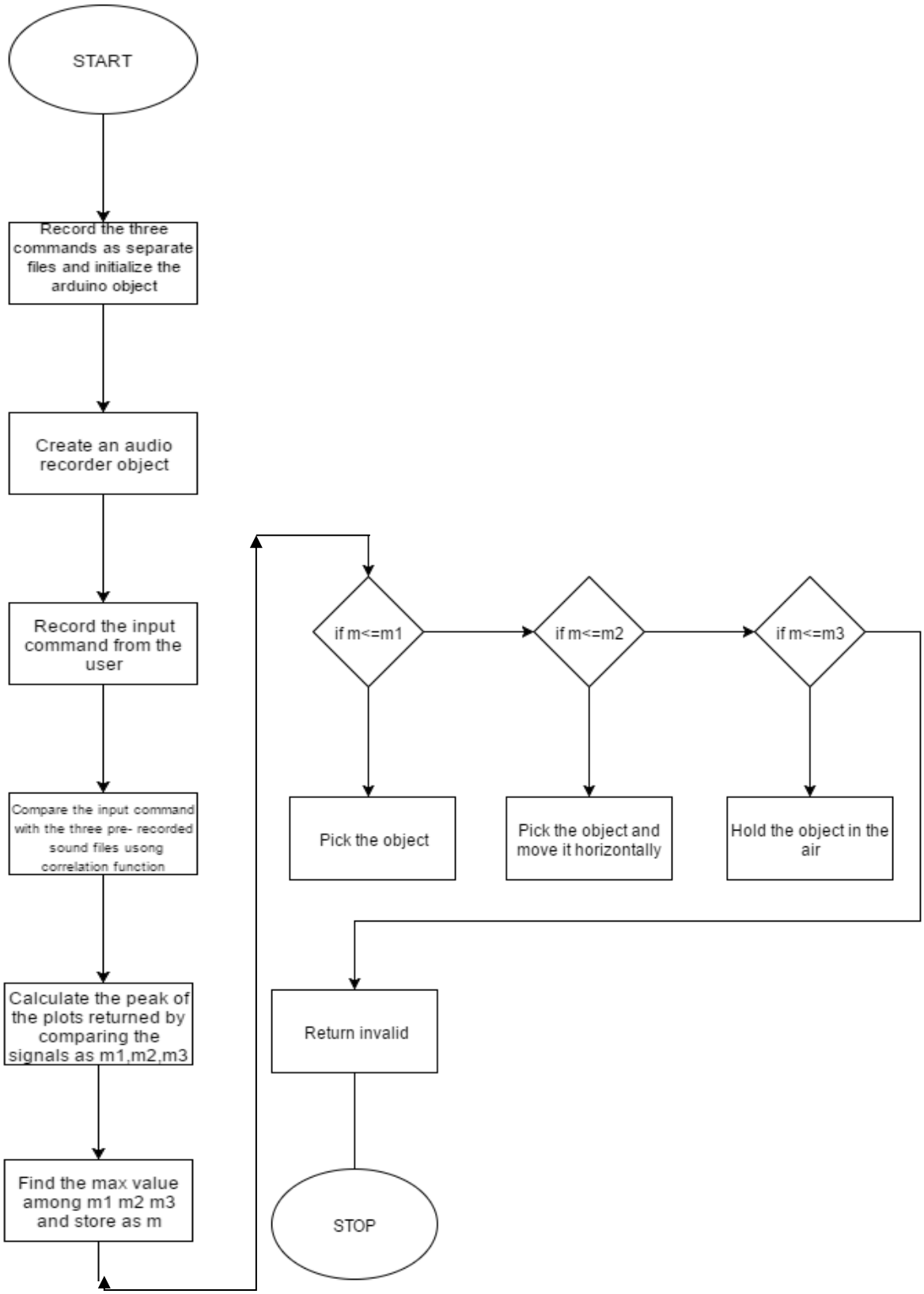


Figure 3.7 Speech Detection Flowchart

Control of Robotic Arm using the GUI

The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user interfaces. We can also create your own custom apps, including their corresponding UIs, for others to use.

The individual motors of the robotic arm can be controlled using the scroll bars given in the GUI. As a result different positions of the arm can be obtained. The GUI also has special preset buttons to perform a few pre-configured gestures.

The processes involved in setting up the GUI include:

1. Declaring the Arduino object and picking and dropping the scroll bars for each motor.
2. guidata(hObject,handles);
   handles.a = get(handles.slider1,'Value')
   The above commands are used to access the position of the slider.
   Where,
   hObject handle to slider
   eventdata reserved - to be defined in a future version of MATLAB
   handles structure with handles and user data (see GUIDATA)
3. The angles are then calibrated on a scale of 0 to 1.
4. Using the writepostion command the servo motors are made to rotate through specific angles to perform the desired action.
5. In the case of preset actions, the push buttons are dragged and placed.
6. The write position commands are specified just under this function block.This causes the arm to perform a specific gesture.

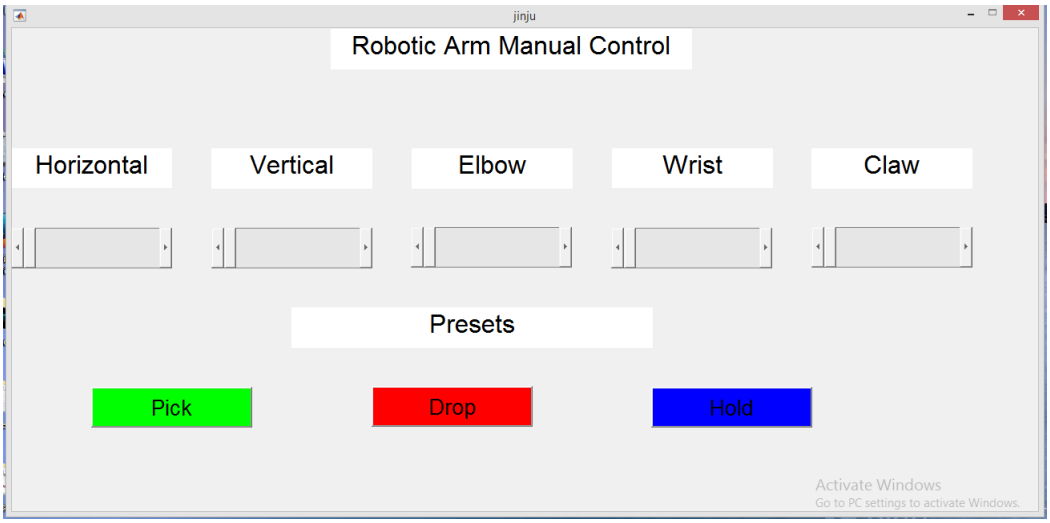


Figure 3.8 Arm Control GUI

Arduino as a server

In this project the arduino is burnt with a code that lets us use Arduino as a server. Matlab can give Arduino commands such as to write a PWM signal or to read an analogue voltage. In this case the no new code is burnt onto the Arduino microcontroller, instead it listens for commands from the Matlab and performs the desired function.

3.4 Summary

This Chapter thus dealt with the processes involved in the construction of the robotic arm and assumptions made with respect to the nature of the inputs and outputs. The section on hardware implementation saw the specifications of the hardware, (i.e) the Arduino Uno and servo motors and the logic behind using them in our project. The software implementation section too described why we had used Matlab and a detailed explanation of the code. Overall block diagrams too were looked at to help understand the essence of the project and get a macroscopic view of how it works.

CHAPTER 4
RESULT ANALYSIS

4.1 Introduction

The robotic arm movement is based on predefined set of movement. The colour detection module works based upon one of the three colours: red, green and blue. Each colour corresponds to a different set of movements. In the same way there are three phrases each corresponding to a different predefined movement. The GUI contains the three predefined movement along with five slider for the movement of the five different motors.

4.2 Result Analysis

Speech Recognition

The speech recognition is done using cross correlation of the pre-recorded signals and the newly recorded signal during runtime.
Three phrases used are:

- Pick the object.
- Drop it down.
- Hold up.

The speech recognition was done for each phrase for 20 times in ambient noise conditions in a random manner.
The following table illustrates the result for each phrase.

Table 4.1 Result Of The Speech Recognition

Phrases	**Correct recognition**	**Incorrect recognition**
Pick the Object	17	2
Drop it down	19	1
Hold up	18	3

From the following table we infer that of the total 60 times the phrases were recorded 54 times the algorithm recognised the phrases correctly and only 6 times there were incorrect recognition of phrases thus achieving a percentage accuracy of 90%.

Colour Recognition

Colour detection is done using separating the three main channels red, green and blue. After the detection of the colour in a frame a corresponding mask is added to the output frame where the colour has been detected. The coloured objects are detected when they are at least 35cm away from the video camera. Darker background coupled with minimal glare gave the best output.

Figure 4.1 Detecting Red Colour

Figure 4.2 Detecting Green Colour

Figure 4.3 Detecting Green Colour

Robotic Arm

The robotic arm movement was governed by the signals sent from the Matlab to the Arduino. Three presets were:

- Picking the object and dropping it down at the same place.
- Picking the object and dropping it down at a new place.
- Picking the object and holding the object up in the air.

These three movement were performed by the arm without dropping the object. The torque provided by the arm is high enough to pick objects up to 200g.
In the theoretical analysis the maximum weight of the object that could be picked by the arm was 100g, but practically it picked the objects without any lag of weights up to 200g.

4.3 Summary

The two algorithms worked satisfactorily when ran in an ambient environment. Colours and speech phrases were recognised with very less error in the output and the preprogramed arm movement performed the desired output without any problem.

CHAPTER 5
CONCLUSIONS & FUTURE SCOPE OF WORK

5.1 Conclusion

In this project a robotic arm was built with four degrees of freedom along with a gripper actuator at the end to pick up the objects.
The arm movement is controlled by a GUI slider where each motor can be moved individually for the desired movement of the arm. Apart from that three predefined movements were also added which did simple tasks such as picking up, dropping and holding the object.
The original goal was to build a gesture based robotic arm which utilised Kinect sensor to track the 3D skeletal joints of the user's arm [4]. The movement of the user's arm would be mimicked by the robotic arm in real time. But the version which was used was not supported by the Matlab. Thus colour sensor and speech recognition module were developed for the control of the robotic arm.
The servo motors used in the robotic arm has higher torque rating than required as per the torque analysis, the torque required for the movement of the arm is much less than what the motors can support. So it can be said that the arm has been overdesigned in terms of the weight it can pick up. Also, the arm can move in all direction as possible by the human hand except the twisting of the forearm and wrist.
The speech recognition program used is not a self-learning program it is simply a comparison between two signals for the likeness between the signals, this causes error as can be seen from the results.
The servo motor used contain position encoder only, thus limiting the possibility to calculate speed. Since it does not have any speed encoder the motor overshoots when it moves a large angular distance, i.e. it exerts a lot of force on the arm structure and increases the possibility of breaking of the arm structure.

5.2 Future Scope

By modelling a human arm in Matlab we can use the model to find out the proper movement of the robotic arm. Modelling of robotic arm will give us more natural and robust movement. Eliminating
By adding more modules to the arm its functionality and usability can be increased. Mounting the arm on a movable arm will give it the ability to move on the ground giving it an extra degree of freedom. Using depth sensing it can detect objects and manoeuvre through a course on its own.

REFERENCES

[1] D.Senthamaraikannan , S.Shriram2 , Dr. J. William "REAL TIME COLOR RECOGNITION", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control engineering (ACDT), 2015.[1]

[2] Amab Pramanik, Rajorshee Raha, Automatic Speech Recognition using correlation analysis, World Congress on Information and Communication Technologies (WICT), 2012.[2]

[3] Krutarth Gandhi, Harsh Kotak, Susmit Joshi & Vikas Pandita, "Motion Controlled Robotic arm", International Journal of Electronics and Communication Engineering (IJECE),2013. [3]

[4] Rajesh Kannan Megalingam, Nihil Saboo, Nitin Ajithkumar, SreeramUnny, Deepansh Menon, "Kinect Based Gesture Controlled Robotic Arm: A Research work at HuT Labs", IEEE, 2013. [4]

ANNEXURES

ANNEXURE A

GUI CODE

```
%MATLAB code for GUI for ROBOTIC ARM

function varargout = jinju(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @jinju_OpeningFcn, ...
   'gui_OutputFcn',  @jinju_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end



function jinju_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes jinju wait for user response (see UIRESUME)

% initialise arduino com port in MATLAB with SERVO variables
clear a;
global a
a = arduino('com3','Uno');
wrist = servo(a,'D10');
elbow = servo(a,'D9');
shoulder_Vert =servo(a,'D6');
shoulder_Hori =servo(a,'D5');
claw = servo(a,'D11');

% --- Outputs from this function are returned to the command line.
function varargout = jinju_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
```

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables

global a
shoulder_Hori = servo(a,'D5');
guidata(hObject,handles);
handles.a = get(handles.slider1,'Value')
b=handles.a
for i = 1:60
   writePosition(shoulder_Hori,b);%change the shoulder horizontal position
end


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables

global a
shoulder_Vert =servo(a,'D6');

guidata(hObject,handles);

handles.a = get(handles.slider2,'Value')

b = handles.a
b = b/180
for i = 1:60
   writePosition(shoulder_Vert,b);%change the shoulder vertical position
end

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
global a
elbow =servo(a,'D9');
guidata(hObject,handles);

handles.a = get(handles.slider3,'Value')
```

```
b=handles.a


for i = 1:60
   writePosition(elbow,b);%change the elbow position
end
% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables

global a
wrist =servo(a,'D10');
guidata(hObject,handles);

handles.a = get(handles.slider4,'Value')

b=handles.a

for i = 1:60
   writePosition(wrist,b);%change the wrist position
end

% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables
global a
claw =servo(a,'D11');
guidata(hObject,handles);

handles.a = get(handles.slider5,'Value')

b=handles.a

for i = 1:60
   writePosition(claw,b);%change the gripper position
end

% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in Green button.
function pushbutton1_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables

global a
wrist = servo(a,'D10');
elbow = servo(a,'D9');
shoulder_Vert =servo(a,'D6');
shoulder_Hori =servo(a,'D5');
claw = servo(a,'D11');

%Predefined motion to pick and drop the object

writePosition(wrist,0.5)
writePosition(elbow,0.5)
writePosition(shoulder_Vert,0.5)
writePosition(claw,0.5)
writePosition(shoulder_Hori,0.5)


for i=1:2

    writePosition(shoulder_Hori,0.5)
    writePosition(wrist,0.7)
    writePosition(elbow,0.6)
    writePosition(shoulder_Vert,0.3)
    writePosition(claw,0.2)
    pause(1)
    writePosition(claw,0.6)
    pause(1)
    writePosition(wrist,0.3)
    writePosition(elbow,0.2)
    pause(2)
    writePosition(wrist,0.7)
    writePosition(elbow,0.6)
    pause(2)
    writePosition(claw,0.2)
    pause(1)

end

% --- Executes on button press in Red button.
function pushbutton2_Callback(hObject, eventdata, handles)
% reinitialise arduino com port with servo variables

global a
wrist = servo(a,'D10');
elbow = servo(a,'D9');
shoulder_Vert =servo(a,'D6');
shoulder_Hori =servo(a,'D5');
claw = servo(a,'D11');
```

```
%Predefined motion to move the object

for i=1:1
    writePosition(claw,0.2)
    pause(1)
    writePosition(wrist,0.6)
    pause(1)
    writePosition(elbow,0.4)
    pause(1)
    writePosition(shoulder_Vert,0.25)
    pause(1)
    writePosition(shoulder_Hori, 0.6)
    pause(1)
    writePosition(claw,0.6)
    pause(1)
    writePosition(shoulder_Hori,0.2)
    pause(1)
   writePosition(claw,0.2)
    pause(1)
    writePosition(shoulder_Hori,0.6)
    pause(1)

end


% --- Executes on button press in Blue t=button.
function pushbutton3_Callback(hObject, eventdata, handles)

% reinitialise arduino com port with servo variables
global a
wrist = servo(a,'D10');
elbow = servo(a,'D9');
shoulder_Vert =servo(a,'D6');
shoulder_Hori =servo(a,'D5');
claw = servo(a,'D11');
%Predefined motion to hold the object
writePosition(claw,0.2)
pause(1)

writePosition(wrist,0.6)
pause(1)

writePosition(elbow,0.4)
pause(1)

writePosition(shoulder_Vert,0.25)
pause(1)

for i = 1:3

    writePosition(shoulder_Hori, 0.6)
    writePosition(claw,0.6)
    pause(2)
    writePosition(shoulder_Vert,0.35)
```

```
    writePosition(elbow,0.2)
    writePosition(wrist,0.25)
    pause(1)
end
```

Speech Recognition

```
% Initialise Arduino COM port and SERVO variables
a = arduino('com3','Uno');

shoulder_Hori = servo(a,'D5');
shoulder_Vert = servo(a,'D6');
elbow = servo(a,'D9');
wrist = servo(a,'D10');
claw = servo(a,'D11');

r = audiorecorder(22050, 16, 1);
fprintf('say a word immediately after hitting enter: ');% Record voice to match
input('');

recordblocking(r,4);
y=getaudiodata(r);

filename = 'govindha.wav';
audiowrite(filename,y,22050);

sound(y,22050)% Play the file that was recorded
pause(3)

speech=audioread(filename);
x=speech;
x=x';
x=x(1,:);
x=x';

y1=audioread('picktheobject.wav');
y1=y1';
y1=y1(1,:);
y1=y1';

z1=xcorr(x,y1);
```

```
m1=max(abs(z1));%Find the cross-correlation between 'picktheobject.wav' and newly recorded audio

y2=audioread('dropitdown.wav');
y2=y2';
y2=y2(1,:);
y2=y2';
z2=xcorr(x,y2);
m2=max(abs(z2));%Find the cross-correlation between 'dropitdown.wav' and newly recorded audio

y3=audioread('holdup.wav');
y3=y3';
y3=y3(1,:);
y3=y3';
z3=xcorr(x,y3);
m3=max(abs(z3));%Find the cross-correlation between 'holdup.wav' and newly recorded audio

m4 = 70;
a=[m1 m2 m3 m4]
m=max(a)

if m<=m1
   soundsc(audioread('picktheobject.wav'),22050)

      fprintf('Picking the object.....')% Perform the gesture for picking the object

      pause(1)

         writePosition(wrist,0.65)
         writePosition(elbow,0.6)
         writePosition(shoulder_Vert,0.3)
         pause(1)
         writePosition(claw,0.2)
         pause(1)
         writePosition(claw,0.6)
         pause(1)
         writePosition(wrist,0.2)
         writePosition(elbow,0.2)
         pause(2)
```

```
        writePosition(wrist,0.65)
        writePosition(elbow,0.6)
        pause(2)
        writePosition(claw,0.2)
        pause(1)

elseif m<=m2
   soundsc(audioread('dropitdown.wav'),22050)% Perform the gesture for dropping the object
     fprintf('Moving and dropping the object.....')

     writePosition(wrist,0.6)
      pause(1)
      writePosition(elbow,0.4)
      pause(1)
      writePosition(shoulder_Vert,0.25)
      pause(1)
      writePosition(shoulder_Hori, 0.6)
      pause(1)
      writePosition(claw,0.2)
      pause(1)
      writePosition(claw,0.6)
      pause(1)
      writePosition(shoulder_Hori,0.2)
      pause(1)
      writePosition(claw,0.2)
      pause(1)
      writePosition(shoulder_Hori, 0.6)
      pause(1)

elseif m<=m3
   soundsc(audioread('holdup.wav'),22050)% Perform the gesture for holding the object

      fprintf('holding the object')


      writePosition(claw,0.2);
      pause(1)

      writePosition(wrist,0.6)
      pause(1)
```

```
        writePosition(elbow,0.4)
        pause(1)

        writePosition(shoulder_Vert,0.25)
        pause(1)

        writePosition(shoulder_Hori, 0.6)
        pause(1)

        writePosition(claw,0.6)
        pause(1)

        writePosition(shoulder_Vert,0.35)
        pause(1)

        writePosition(elbow,0.2)
        pause(1)

        writePosition(wrist,0.25)
        pause(1)
elseif m<=m4
  fprintf('Invalid')% Print invalid if none of the audio files match
end
```

Colour Detection

```
clc; clear;
% Initialise Arduino COM port and SERVO variables
a = arduino('com3','Uno');
wrist = servo(a,'D10');
elbow = servo(a,'D9');
shoulder_Vert = servo(a,'D6');
shoulder_Hori = servo(a,'D5');
claw = servo(a,'D11');
%Initial position
writePosition(claw,0.2);
writePosition(wrist,0.6);
writePosition(elbow,0.4)
writePosition(shoulder_Vert,0.25)
writePosition(shoulder_Hori, 0.6)

%Threshold for each colour channel
```

```
redThresh = 0.24;
greenThresh = 0.05;
blueThresh = 0.15;

vidDevice = imaq.VideoDevice('winvideo', 1, 'YUY2_640x480','ROI', [1 1 500 300], ...
   'ReturnedColorSpace', 'rgb'); %initialise video device

vidInfo = imaqhwinfo(vidDevice);


hblob = vision.BlobAnalysis(    'AreaOutputPort',       false, ...
   'CentroidOutputPort',    true, ...
   'BoundingBoxOutputPort', true', ...
   'MinimumBlobArea',       5000, ...
   'MaximumBlobArea',       10000, ...
   'MaximumCount',          1); %initialise blob analysis for counting the detected colours


hshapeinsBox = vision.ShapeInserter('BorderColorSource', 'Input port', ...
   'Fill', true, ...
   'FillColorSource', 'Input port', ...
   'Opacity', 0.4); % To insert the mask

htextinsRed = vision.TextInserter('Text', 'Red   : %2d', ...
   'Location',  [5 2], ...
   'Color', [1 0 0], ...
   'Font', 'Courier New', ...
   'FontSize', 14);

htextinsGreen = vision.TextInserter('Text', 'Green : %2d', ...
   'Location',  [5 18], ...
   'Color', [0 1 0], ...
   'Font', 'Courier New', ...
   'FontSize', 14);

htextinsBlue = vision.TextInserter('Text', 'Blue  : %2d', ...
   'Location',  [5 34], ...
   'Color', [0 0 1], ...
   'Font', 'Courier New', ...
   'FontSize', 14);

htextinsCent = vision.TextInserter('Text', '+      X:%4d, Y:%4d', ...
   'LocationSource', 'Input port', ...
   'Color', [1 1 0], ...
   'Font', 'Courier New', ...
   'FontSize', 14);
% Insert the box with each colour showing the number of counts available

hVideoIn = vision.VideoPlayer('Name', 'Final Video', ...
   'Position', [100 100  vidInfo.MaxWidth+20 vidInfo.MaxHeight+30]);
```

```
nFrame = 0;


while(nFrame < 5000)
   rgbFrame = step(vidDevice);
   rgbFrame = flipdim(rgbFrame,2);

   diffFrameRed = imsubtract(rgbFrame(:,:,1), rgb2gray(rgbFrame)); % Extract the red
channel
   diffFrameRed = medfilt2(diffFrameRed, [3 3]); % Filter the image
   binFrameRed = im2bw(diffFrameRed, redThresh); % Get the area where mask will be put

   diffFrameGreen = imsubtract(rgbFrame(:,:,2), rgb2gray(rgbFrame)); % Extract the green
channel
   diffFrameGreen = medfilt2(diffFrameGreen, [3 3]);% Filter the image
   binFrameGreen = im2bw(diffFrameGreen, greenThresh);% Get the area where mask will
be put

   diffFrameBlue = imsubtract(rgbFrame(:,:,3), rgb2gray(rgbFrame));% Extract the blue
channel
   diffFrameBlue = medfilt2(diffFrameBlue, [3 3]);% Filter the image
   binFrameBlue = im2bw(diffFrameBlue, blueThresh);% Get the area where mask will be
put

   [centroidRed, bboxRed] = step(hblob, binFrameRed);
   centroidRed = uint16(centroidRed);

   [centroidGreen, bboxGreen] = step(hblob, binFrameGreen);
   centroidGreen = uint16(centroidGreen);

   [centroidBlue, bboxBlue] = step(hblob, binFrameBlue);
   centroidBlue = uint16(centroidBlue);
   % initialise variable to put the centroid values
   rgbFrame(1:50,1:90,:) = 0;
   vidIn = step(hshapeinsBox, rgbFrame, bboxRed, single([1 0 0]));
   vidIn = step(hshapeinsBox, vidIn, bboxGreen, single([0 1 0]));
   vidIn = step(hshapeinsBox, vidIn, bboxBlue, single([0 0 1]));

   for object = 1:1:length(bboxRed(:,1));
      centXRed = centroidRed(object,1); centYRed = centroidRed(object,2);
      vidIn = step(htextinsCent, vidIn, [centXRed centYRed], [centXRed-6 centYRed-9]);
   end

   for object = 1:1:length(bboxGreen(:,1));
      centXGreen = centroidGreen(object,1);
      centYGreen = centroidGreen(object,2);
      vidIn = step(htextinsCent, vidIn, [centXGreen centYGreen], [centXGreen-6
centYGreen-9]);
   end
```

```
    for object = 1:1:length(bboxBlue(:,1));
       centXBlue = centroidBlue(object,1); centYBlue = centroidBlue(object,2);
       vidIn = step(htextinsCent, vidIn, [centXBlue centYBlue], [centXBlue-6 centYBlue-9]);
    end


    vidIn = step(htextinsRed, vidIn, uint8(length(bboxRed(:,1))));
    vidIn = step(htextinsGreen, vidIn, uint8(length(bboxGreen(:,1))));
    vidIn = step(htextinsBlue, vidIn, uint8(length(bboxBlue(:,1))));

    step(hVideoIn, vidIn);
    nFrame = nFrame+1;

    br= length(bboxRed(:,1)) % Get the count for red colour
    bg= length(bboxGreen(:,1)) % Get the count for red colour
    bb= length(bboxBlue(:,1)) % Get the count for red colour

    if (bb>0)
% perform the gesture to hold the object
      fprintf('Holding the Object.....')
        writePosition(claw,0.2);
        pause(1)
        writePosition(wrist,0.6)
        pause(1)
        writePosition(elbow,0.4)
        pause(1)
        writePosition(shoulder_Vert,0.25)
        pause(1)
        writePosition(shoulder_Hori, 0.6)
        pause(1)
        writePosition(claw,0.6)
        pause(1)
        writePosition(shoulder_Vert,0.35)
        pause(1)
        writePosition(elbow,0.2)
        writePosition(wrist,0.25)
        pause(1)

    end

    if (bg > 0)

        fprintf('Lifting the Object.....')
        % perform the gesture to lift the object
        for i=1:2
           pause(1)

           writePosition(wrist,0.65)
           writePosition(elbow,0.6)
```

```
            writePosition(shoulder_Vert,0.3)
            pause(1)
            writePosition(claw,0.2)
            pause(1)
            writePosition(claw,0.6)
            pause(1)
            writePosition(wrist,0.2)
            writePosition(elbow,0.2)
            pause(2)
            writePosition(wrist,0.65)
            writePosition(elbow,0.6)
            pause(2)
            writePosition(claw,0.2)
            pause(1)
        end
end

if(br>0)
    fprintf('Moving the Object.....')

        % perform the gesture to move the object
        writePosition(wrist,0.6)
        pause(1)
        writePosition(elbow,0.4)
        pause(1)
        writePosition(shoulder_Vert,0.25)
        pause(1)
        writePosition(shoulder_Hori, 0.6)
        pause(1)
        writePosition(claw,0.2)
        pause(1)
        writePosition(claw,0.6)
        pause(1)
        writePosition(shoulder_Hori,0.2)
        pause(1)
        writePosition(claw,0.2)
        pause(1)
        writePosition(shoulder_Hori, 0.6)
        pause(1)
    end

end

release(hVideoIn);
release(vidDevice);
clear all;
clc;
```

PROJECT DETAILS

Student Details			
Student Name	**Ananth S**		
Register Number	120906758	Section / Roll No	D / 45
Email Address	Ananth594@gmail.com	Phone No (M)	9740614208
Student Name	**Rahul Yadav**		
Register Number	120906832	Section / Roll No	D / 53
Email Address	Rahul.yadav.18@live.com	Phone No (M)	8867498222
Project Details			
Project Title	**GUI based Arduino controlled Robotic Arm using Colour detection and Speech recognition**		
Project Duration	4 months	Project Start Date	11–01–2016
Organization Details			
Organization Name	**Manipal Institute of Technology, Manipal**		
Full postal address with pin code	Dept of Electrical & Electronics Engg, MIT Manipal, Manipal 576 104, Karnataka State, INDIA		
Guide Details			
Guide Name	**Dr. B. K. Singh**		
Full contact address with pin code	Dept of Electrical & Electronics Engg, MIT Manipal, Manipal 576 104, Karnataka State, INDIA		
Email address	Bk.singh@manipal.edu		