A REPORT
ON
ROBOTICS PROJECT
BY

Name(s) of the Students(s)	ID.No. (s)
NIKHIL AGRAWAL	2016A4PS0248P
CH RAM PRAVEEN	2016A4PS0359H
VARUN GORADIA	2016A8PS0433P
V SAI SUDHIR	2016A8PS0386H
SHRIKUNJ SARDA	2016B1A80812P
GARVIT TANEJA	2016B2A80911P
PRANIT BAVISHI	2016B3A30332P

AT
US TECHNOLOGY GLOBAL
A Practice School-1 Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(May, 2018)

A REPORT
ON
ROBOTICS PROJECT

BY

Name(s) of the Students(s)	ID.No. (s)	Discipline(s)
NIKHIL AGARWAL	2016A4PS0248P	B.E, MECH
CH RAM PRAVEEN	2016A4PS0359H	B.E, MECH
VARUN GORADIA	2016A8PS0433P	B.E, ENI
V SAI SUDHIR	2016A8PS0386H	B.E, ENI
SHRIKUNJ SARDA	2016B1A80812P	B.E, ENI
GARVIT TANEJA	2016B2A80911P	B.E, ENI
PRANIT BAVISHI	2016B3A30332P	B.E, EEE

Prepared in partial fulfillment of the

Practice School-1 Course No.

BITS F221

AT

US TECHNOLOGY GLOBAL

A Practice School-1 Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(May, 2018)

ACKNOWLEDGEMENTS

The opportunity of Practice School 1 at **UST Global, Trivandrum** served as a great platform for learning and developing skills at such an esteemed organization. Hence, we would like to thank Sajan Pillai, CEO of UST Global and Practice School Division, BITS Pilani for granting us this opportunity. We express our utmost gratitude towards the two organizations

We would like to thank our PS instructor, Dr. Sudha Radhika who gave us all necessary advices, guidance and arranged every facility to help us get the best of everything during our PS program. We are also indebted to our project mentor, Mr. Ashok G Nair, Director of Delivery services for extending his support and guidance in spite of his hectic schedule.

We are thankful to Mrs.Rajalakshmi Babu Rajan for providing the desired projects by knowing our interests. We express gratitude to the HR Manager Mrs. Revathi Srinivas for helping us during the orientation program explaining the corporate culture. We are thankful to Mr. Rajesh Kumar for patiently addressing our problems and providing us everything we require. Finally, we would thank the staff of UST who made our stay comfortable and our fellow BITSians who were always there for help and suggestions.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

Practice School Division

Station: US Technology Global Centre: Trivandrum

Duration: 53 Days Date of start: 22nd May, 2018

Date of Submission: 9th July, 2018

Title of the Project: 3D Modeling & analysis and implementation of object Identification, gesture and voice controls in Robotic arm.

Name(s) of the Students(s)	ID.No. (s)	Discipline(s)
NIKHIL AGARWAL	2016A4PS0248P	B.E, MECH
CH RAM PRAVEEN	2016A4PS0359H	B.E, MECH
VARUN GORADIA	2016A8PS0433P	B.E, ENI
V SAI SUDHIR	2016A8PS0386H	B.E, ENI
SHRIKUNJ SARDA	2016B1A80812P	B.E, ENI
GARVIT TANEJA	2016B2A80911P	B.E, ENI
PRANIT BAVISHI	2016B3A30332P	B.E, EEE

Name and Designation of the expert: ASHOK.G.NAIR
Director, Product Delivery Services

Name of the PS faculty: DR. SUDHA RADHIKA
BITS Pilani, Hyderabad.

Key Words: CAD, Finite Element Method, end effector, Arduino, Google Voice kit, OpenCV, Python, SPI Protocol, GPIO, I2C Protocol,

Project Areas: Robotics, Design and analysis, Electronics.

ABSTRACT

In this project, we will design 6 DOF Robotic arm in Solid works, static analysis of each individual part in Ansys and implementation of a kinematic model for 6 DOF robot arm is developed, model robot performance can be checked numerically using results from coordinate's frames, which set the proposed matrices by Denavit-Hartemberg method to determine the robot joins angle vector. Both forward and reverse kinematic analysis will be done.

Machine Vision enables computer to recognize and evaluate images. It enables a computing device to inspect, evaluate and identify still or moving images. We want to achieve object detection through OpenMV M7 camera that has a micro python library as well as IDE for programming it. We discussed 5 methods of object detection and decided among them on the basis of processing time, accuracy and complexity of the algorithms. After discussing all the methods we came to a conclusion that using CMSIS-NN library for training a caffe model and deploying onto the camera that is on the SD-card than running the model on the camera is the best method as it is the most suitable among all.

The gesture control project is a real time monitoring system by which humans interact with robots through gestures. The implementation is achieved by navigation of robot through various gestures. The gesture control is implemented in such a way that one can control not only the arm but also any kind of robot which runs on motors. The gestures implemented are very simple and do not involve any complex gestures. By the impact of the project, life of physically challenged people becomes less challenging. Also, many applications that are mentioned in the report can be implemented.

Speech Recognition will revolutionize the way people interacted with smart devices and will ultimately differentiate the upcoming technologies. Almost all the smart devices coming today in the market are capable of recognizing speech. Many areas can benefit from this technology. Speech Recognition can be used for intuitive operation of computer-based systems in daily life.

Signature(s) of the Student(s) Signature of the PS Faculty

Date: 13th July, 2018 Date:

TABLE OF CONTENTS

ABOUT UST GLOBAL……9

INTRODUCTION……10

DESIGN AND ANALYSIS…….....11

6 DOF ARM MODELING……11

STEPS FOR MODELING……..12

PARTS OF ROBOTIC ARM……..12

TORQUE CALCULATIONS…….17

STATIC STRUCTURAL ANALYSIS……..19

NEED……..19

FINITE ELEMENT METHOD (FEM)……..20

STEPS IN FEM……...20

FEM IN ANSYS…….....20

DEFORMATION AND STRESS IN ANSYS……...21

GRIPPER ROTOR ……21

BASE…….22

SERVO HOLDER…….23

ARM HOLDER…….25

JOINT HOLDER……...26

CONNECTING LINK……...27

SERVO GEAR……..28

KINEMATIC ANALYSIS OF ROBOTIC ARM…….29

FORWARD KINEMATIC ANALYSIS……....29

DEFINITION……29

ALGORITHM……...29

NUMBERING OF LINKS……29

DH PARAMETERS……...30

DH AXIS DIAGRAM……...32

EQUATIONS……33

INVERSE KINEMATIC ANALYSIS…….37

DEFINITION……....37

PIPER SOLUTION……...37

GEOMETRICAL SOLUTION……37

ANALTICAL SOLUTION…….....39

MACHINE VISION…….....43
INTRODUCTION……...43
MACHINE VISION SYSTEM…….....43
SENSING DATA……43
PROCESSING IMAGE……...44
HARDWARE……..45
OPENMV M7 CAMERA……...45
INSTALL CAMERA ON ARM……46
METHODS OBJECT DETECTION……..48
USING OBJECT DETECTION API……..48
SURF+FLANN……...49
FEATURE EXTRACTION WITH SURF……..49
FEATURE MATCHING WITH FLANN……...50
CONOTUR+DESCRIPTOR EXTRACTION…….....50
CONTOUR EXTRACTION……..……50
DESCRIPTOR CONCEPT……51
PROPOSED ALGORITHM……..52
USING BUITL IN LIBRARIES OF IDE……...….52
CMSIS-NN LIBRARY……54
CONCLUSION……55
GESTURE CONTROL……57
INTRODUCTION…….....57
HARDWARE……..57
ARDUINO……...……58
NODEMCU AND WI-FI ROUTER……...….60
OTHER ALTERNATIVE……...……62
RF TRANSCEIVER……...……...62
HT-12E AND HT-12D……..…62
MPU6050……..…63
ROBOTIC ARM……..….64
OTHER INTERFACING HARDWARE……..…64
HARDWARE CONNECTIONS……..…65
SENDER CIRCUIT……..65
RECEIVER CIRCUIT……..……65
CONNECTION DESCRIPTIONS……..……66
HARDWARE COST OUTLOOK……..67

ADDITIONAL HARDWARE FOR ARM GRIPPER……..68
SOFTWARE……..69
WORKING ALGORITHMS…….70
SENDER ALGORITHM……70
RECEIVER ALGORITHM……70
GESTURES FOR ARM MOTION……...…71
STOP, FORWARD AND BACKWARD GESTURES…….....71
LIFT AND DROP GESTURES…….72
MOVE RIGHT AND MOVE LEFT GESTURES…….73
ADVANTAGES……73
DISADVANTAGES……..73
REAL LIFE APPLICATIONS……...74
VOICE RECOGNITION……75
INTRODUCTION…….75
SPEECH RECOGNITION ALGORITHMS…….76
DYNAMIC TIME WARPING…….....76
HIDDEN MARKOV MODEL……...76
WHAT IS A MARKOV MODEL? ……....77
NEURAL NETWORKS……...78
SPEECH RECOGNITION HARDWARE……...79
IN THE KIT……..80
STEPS TO BE FOLLOWED……81
ADVANTAGES OF SPEECH RECOGNITION…….82
CONCLUSION…….82
APPENDIX……...83
REFERENCES…….99
GLOSSARY…….....100

ABOUT UST GLOBAL:

UST Global® is a leading digital technology company that provides advanced computing and digital services to large private and public enterprises around the world. It is a leading provider of end-to-end IT services and solutions for Global 1000 companies. They use a client-centric Global Engagement Model that combines local, senior, on-site resources with the cost, scale, and quality advantages of off-shore operations.

The industry-leading expertise found within our Centers of Excellence (CoEs) plays a key role in their success with clients. The CoEs deliver pragmatic IT solutions that allow clients to consistently achieve their most critical business objectives. The program aims to develop strategic relationships with best-of-breed organizations to provide UST Global developers with advance access to new technology and educational resources.

The following services are provided by the company:

1. Analytics
2. Cloud
3. Cyber security
4. Information management
5. Infrastructure services
6. IOT
7. Legacy Modernization
8. Mobility
9. Portfolio assessment

They provide their services to Banking, manufacturing, retail, media & entertainment, semiconductor, healthcare, public sector, telecommunications and High tech industries.

Some of their major clients include Amazon, Cisco, IBM, Kony, Vidyo, SAP, Oracle, FICO etc.

One of the major divisions of the company is Infinity Labs. It is a network of innovation gyms where professionals are invited to experiment without fear of mistakes. The main functionality of the division includes

1. Research lab for developing user cases for new technologies.
2. Solving business and technology problems for the company's Retail, CPG and Logistics clients.
3. Giving clients an immersive experience of UST Global's capabilities and services.

INTRODUCTION:

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm, the arm may be the sum total of the mechanism or may be part of a more complex robot. Most robots in the world are designed for heavy, repetitive manufacturing work. They handle tasks that are difficult, dangerous or boring to human beings. The most common manufacturing robot is the robotic arm.

The robotic arm given for experimentation as a prototype is U-Arm Swift Pro. It is a Cartesian SCARA (Selective Compliance Articulated Robotic Arm) type robot used for pick and place work, application of sealant, assembly operations and handling machine tools. This robot features two parallel rotary joints to provide compliance in a plane.

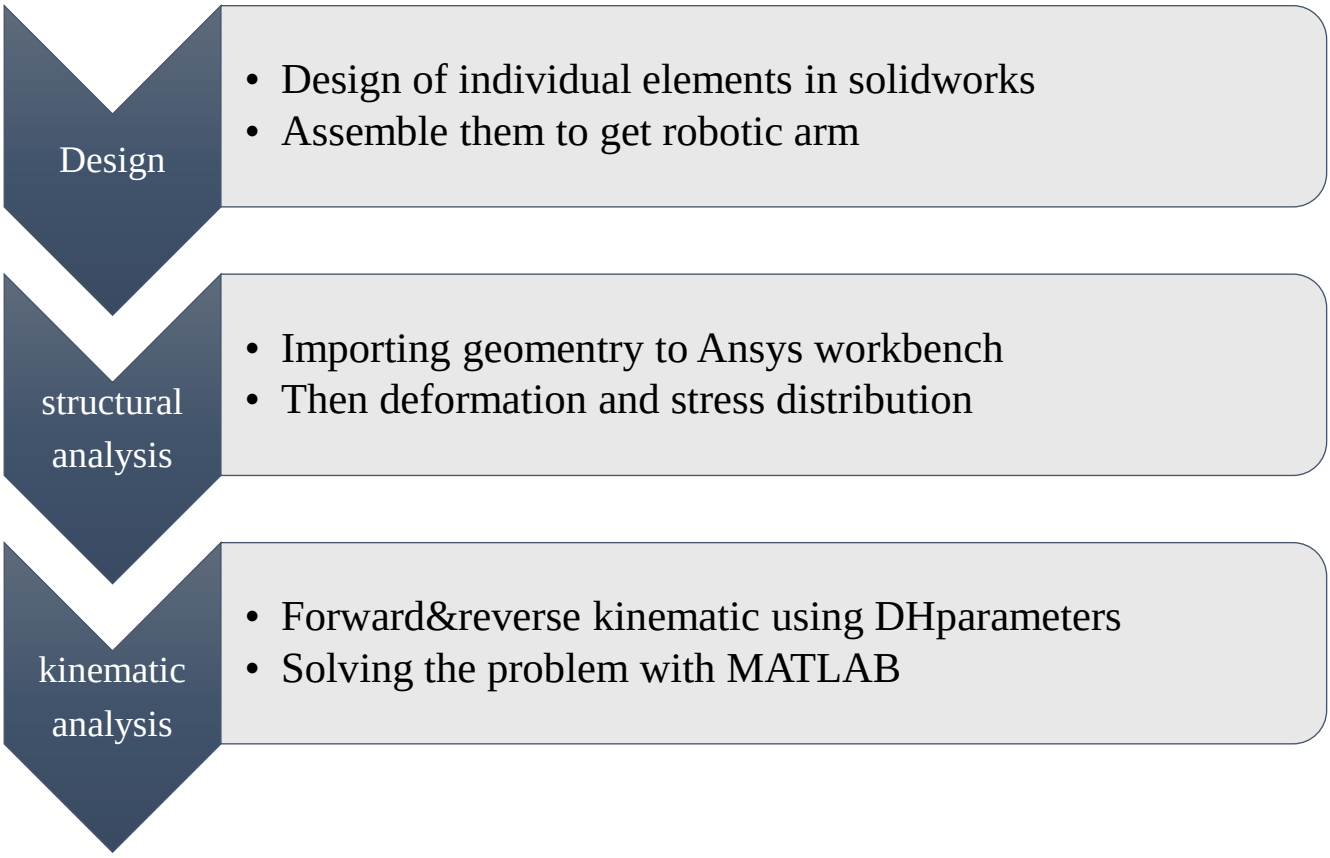
There are five innovative ideas to implement in a robotic arm, they are:

(1) Object Identification
(2) Gesture control
(3) Voice command control
(4) Posture
(5) Stereo vision

Apart from these, 3D CAD modelling and analysis of robotic arm is also a part of the project.

MAIN AIM-The aim of the project is to program a robotic arm to detect, pick the objects from the conveyer belt and place them onto an instructed position. The five tasks discussed above together helps us to build the project.

PROJECT DESCRIPTION:



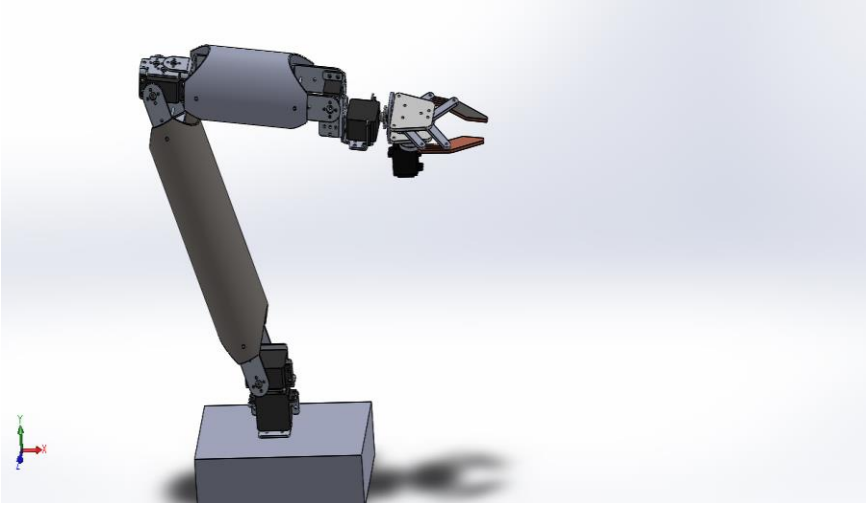
6 DOF CAD model of Robotic arm:

The most important aspect of this project is the mechanical design of the robotic arm. A robotic arm has certain design specifications and certain parameters are to be taken into the consideration. The design specifications which I was given are 6 DOF and maximum payload 20kg.

The CAD software, SolidWorks is used to model the design of elements of robotic arm and assembly of parts to make a robotic arm. The robotic arm used six servomotors for overall operation; five for its joints and one for the gripper mechanism

There is a servo motor at the base of the arm, which allows for angular movement of the whole structure; other two at the shoulder and elbow to allow the upward and downward movement of the arm; two for the movement of the wrist while the last servo motor at the end effector allows for the gripping of

objects. Servo motors serve as the actuators at various joint. The motors are controlled by Arduino microcontroller upon receiving commands

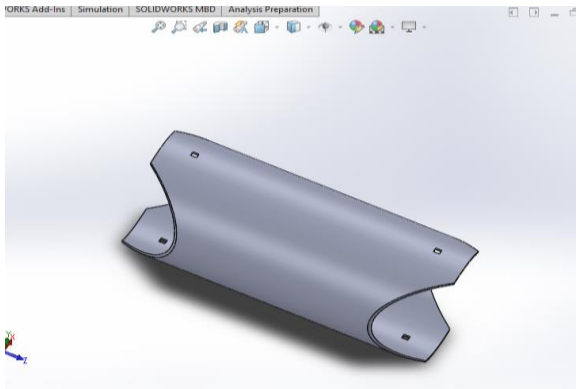


Steps for modelling a robotic arm

- Examine the blue print of the robotic arm.
- Then mark the required deimentions then draw the sketch of individual element and extrude it.
- After modelling all the individual parts then join them using assemble component feature in solid works.

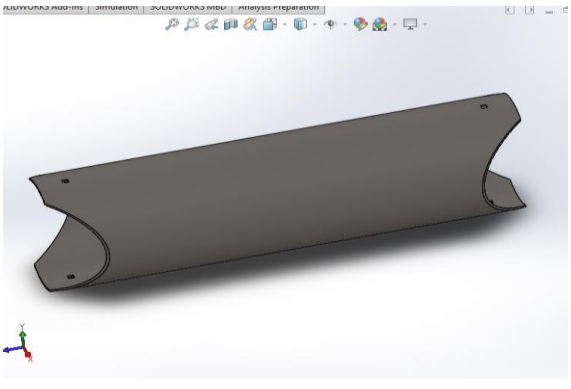
Components of a robotic arm:

Small arm



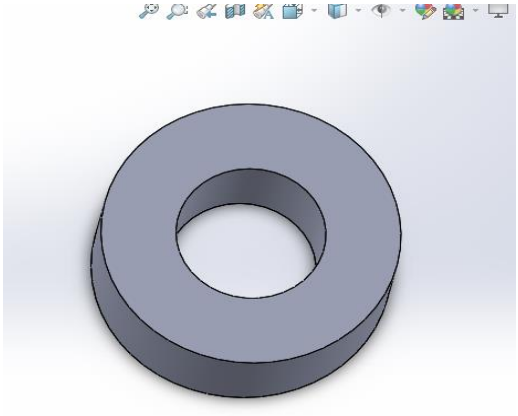
Dimensions
Diameter -65mm
Length -150mm

Large arm

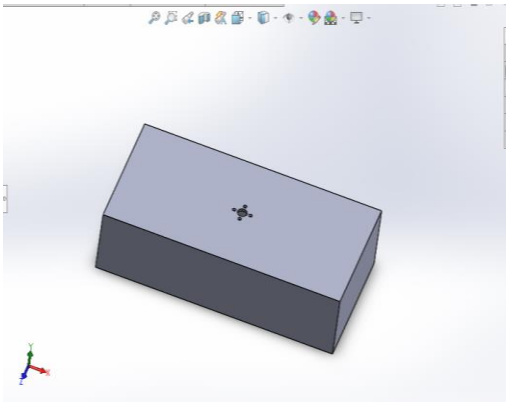


Dimensions
diameter-65mm
length -250mm

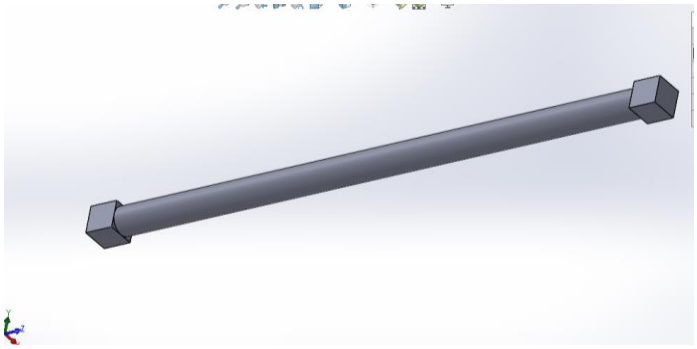
Joint holder



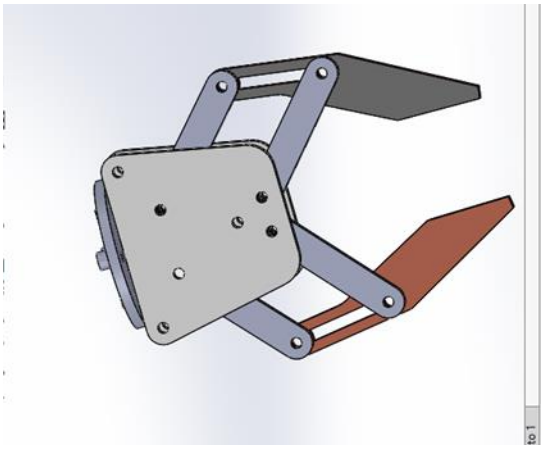
Arm base



Support rod

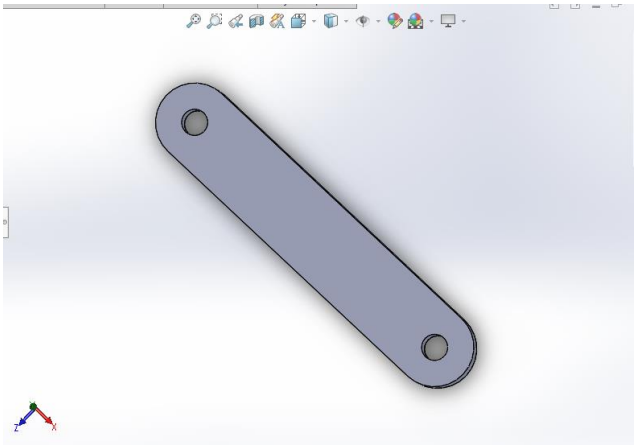


Gripper:

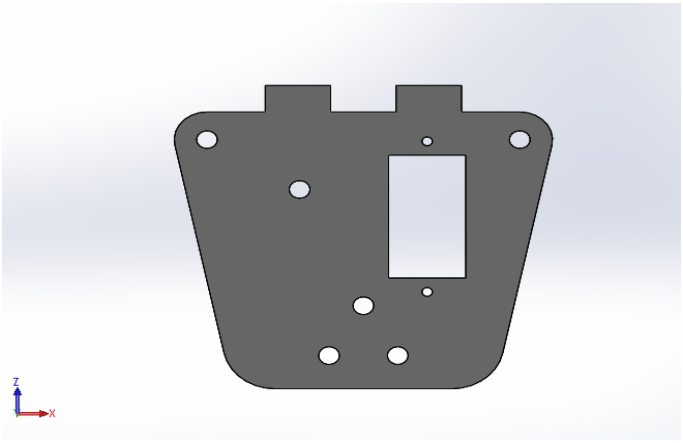


Gripper parts:

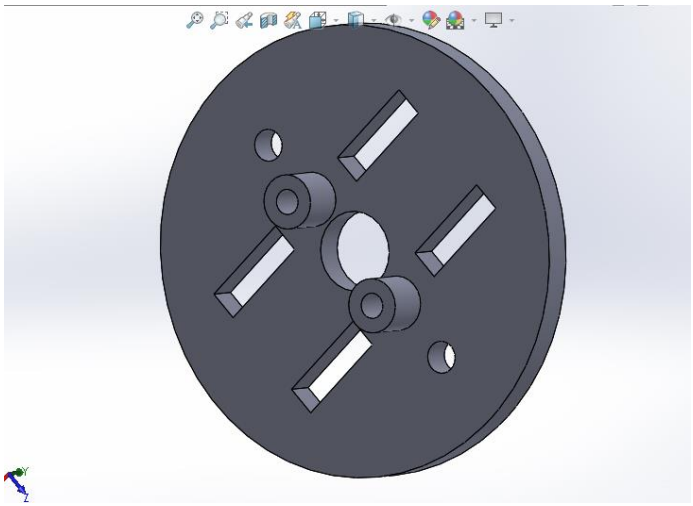
Connecting link



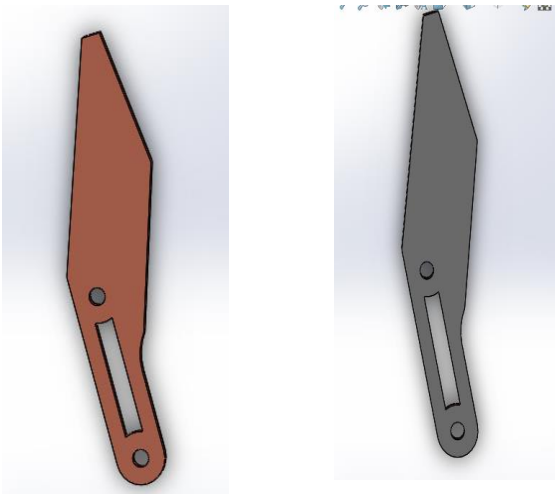
Gripper slide plate-1



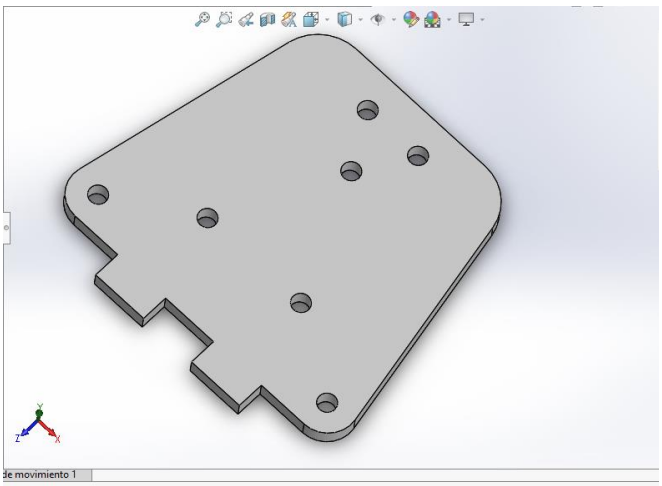
Gripper base



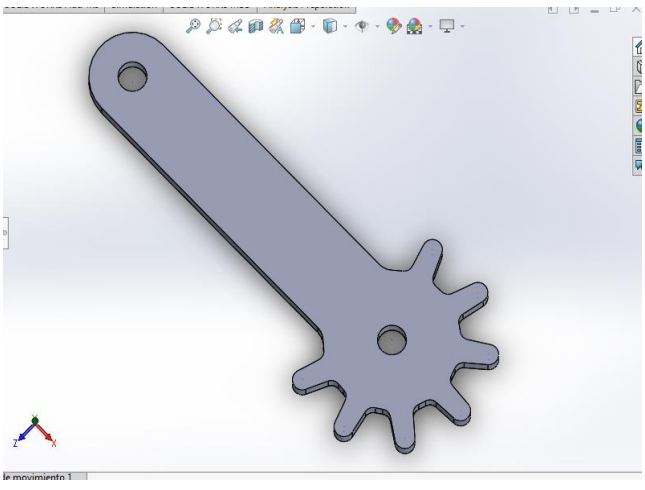
Gripper claws



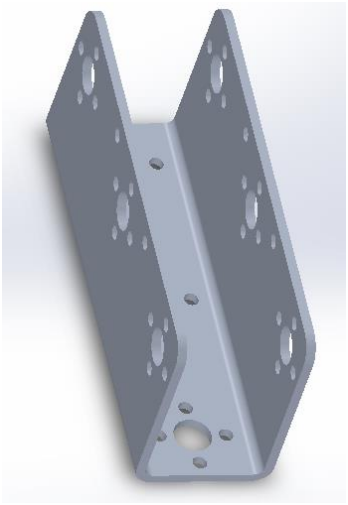
Gripper side plate-1

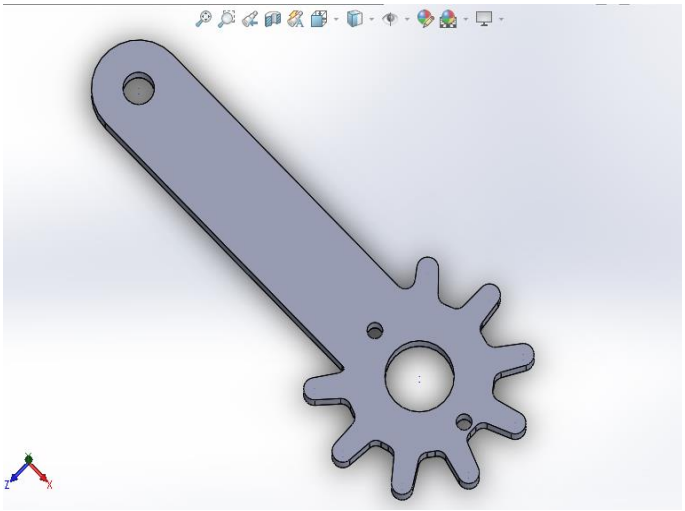


Gear mechanism link-1



Servo holders:

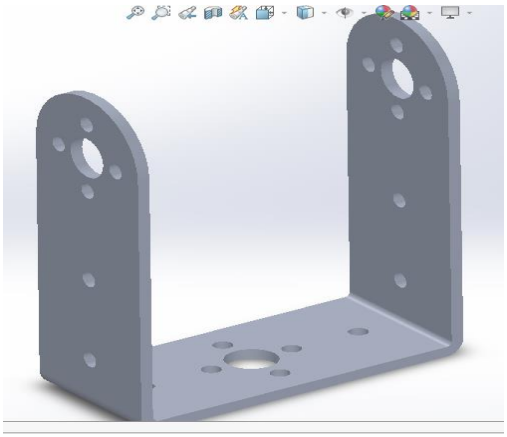


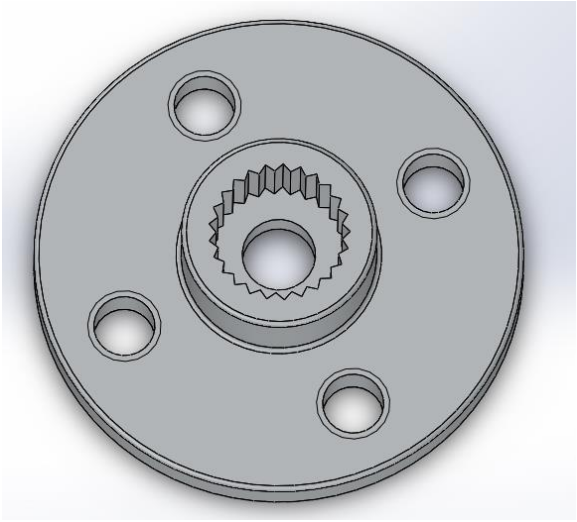


Gripper mechanism link-2

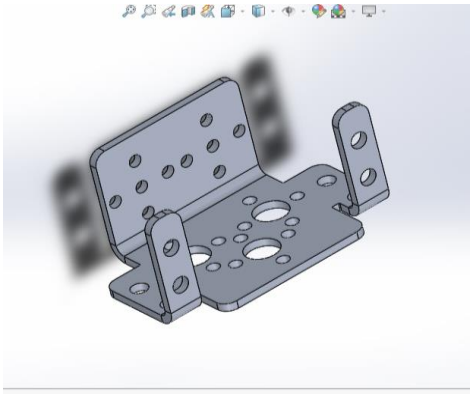
Servo motor

Arm holders:

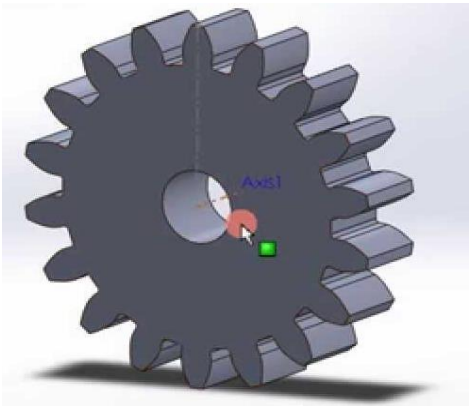




Gripper rotator



Servo base plate



Servo gear

Torque caluculations of robotic arm:

Torque is measure of how much force is acting on an object and makes that object to rotate. It is denoted by T.

Torque (T) is defined as a moving "force" and is calculated using the following equation:

T=F*L … (1),where T is torque F means calculated force and L is denoted the length from a pivot point.

The force is accelerating on an object due to gravity (g = 9.81m/s2) multiplied by its mass.

F=M*g... (2),where Mass (M) and gravity (g)

The force (F) is also considered of an object's weight (W)

W=M*g... (3)

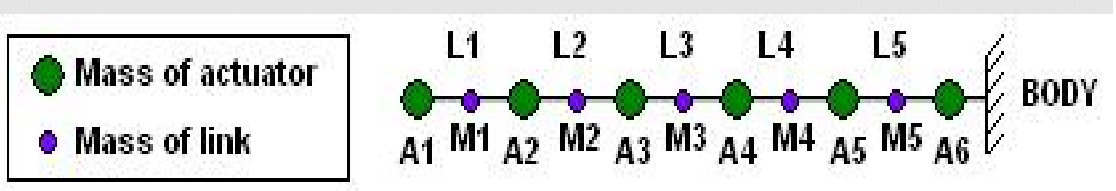
The torque required to hold a mass at a given distance from a pivot point is showing therefore

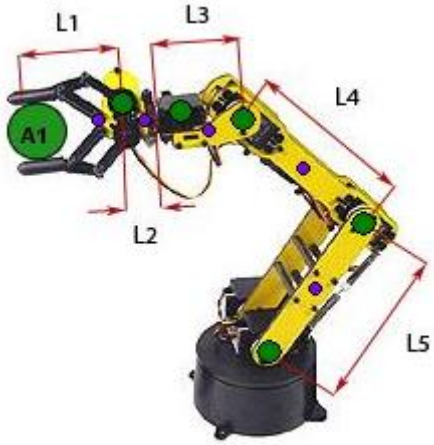
T=(M*g)*L… (4)

The length L is the perpendicular length from a pivot point to the force. This equation can found by similar doing a torque balance about a point.

$\sum$**T = 0 = F*L – T**.. (5)

Therefore, replacing the force (F) with mass and gravity (m*g) we can find out the same equation above.
This is the more accurate way to find out the torque by using the torque balance.
M*g*L = T_A
For the robotic arm,





$$T1 = L1 * A1 + \frac{1}{2} L1 * W1$$

$$T2 = (L1 + L3) * A1 + (\frac{1}{2} L1 + L3) * W1 + (L3) * A2 + (\frac{1}{2} L3) * W2$$

Like this we can calculate torques for different joints and it will get more complecated with equations as we proceed on.So it is better to use web GUI for calculating these torqes

L: [cm]		M: [kg]		A: [kg]		T: [kg cm]	
L1:	15	M1:	0.2	A1:	0.01	T1:	1.65
L2:	5	M2:	0.05	A2:	0.01	T2:	2.875
L3:	10	M3:	0.3	A3:	0.01	T3:	7.175
L4:	50	M4:	0.5	A4:	0.01	T4:	49.17
L5:	60	M5:	0.6	A5:	0.01	T5:	133.1

With this we can know how much torque is required for different joints to move based on that we can place the required servo motor which has the power to rotate that joint.This method is useful to find out the required rpm of sevo motor.

Static structural analysis in Ansys:

In static structural analysis we actually determine the displacements, stresses, strains, and forces in structures or components which are caused by loads that do not induce damping effects and significant inertia forces. Steady loading and response conditions are assumed i.e. the loads and the structure's response are expected to vary slowly with respect to time. A static structural load can be performed using any simulation software. Here I am using Ansys software.

The types of loading that can be applied in a static analysis include:

- Externally applied forces and pressures
- Steady-state inertial forces (such as gravity or rotational velocity)
- Imposed (nonzero) displacements
- Temperatures (for thermal strain)

Need for structural analysis:

A good design is always judged by its load bearing capability. In operations we expect them to be stronger than the essential, so that they can perform the necessary action for a long time. In engineering, stress analysis is often a tool rather than a goal; the ultimate goal being the design of structures that can withstand a specified load, using the minimum amount of material or that satisfies some other optimality criterion. The stress on the part can be related with the load carrying capacity. The result of the analysis is a description of how the applied forces spread throughout the structure, resulting in stresses, strains and the deflections of the entire structure and each component of that structure. The analysis may consider forces that vary with time, such as engine vibrations or the load of moving vehicles.

The results of the analysis are used to verify a structure's fitness for use, often preventing physical tests. As a result, we can decide whether the design can sustain in physical world. This method is economical because we can manufacture the efficient product. Structural analysis employs the fields of applied mechanics, materials science and applied mathematics to compute a structure's deformations, internal forces, stresses, support reactions, accelerations, and stability. Research organizations like ISRO, NASA are simulating their rockets before launching them.

Finite element analysis:

The finite element method (FEM) is a numerical approach for solving problems of engineering and mathematical physics. Typical problem areas of interest include structural analysis, heat transfer, fluid flow, mass transport, dynamic analysis and electromagnetic potential. The analytical solution of these problems generally requires the solution to boundary value problems for partial differential equations.

The finite element method formulation of the problem results in a system of algebraic equations. The method yields approximate values of the unknowns at discrete number of points over the domain. To solve the problem, it subdivides a large problem into smaller, simpler parts that are called finite elements. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. FEM then uses variational methods from the calculus of variations to approximate a solution by minimizing an associated error function.

Steps for FEM analysis are

1. Pre-analysis
2. Geometry
3. Mesh
4. Model setup
5. Numerical solution
6. Numerical results
7. Verification and validation

Finite element analysis in Ansys:

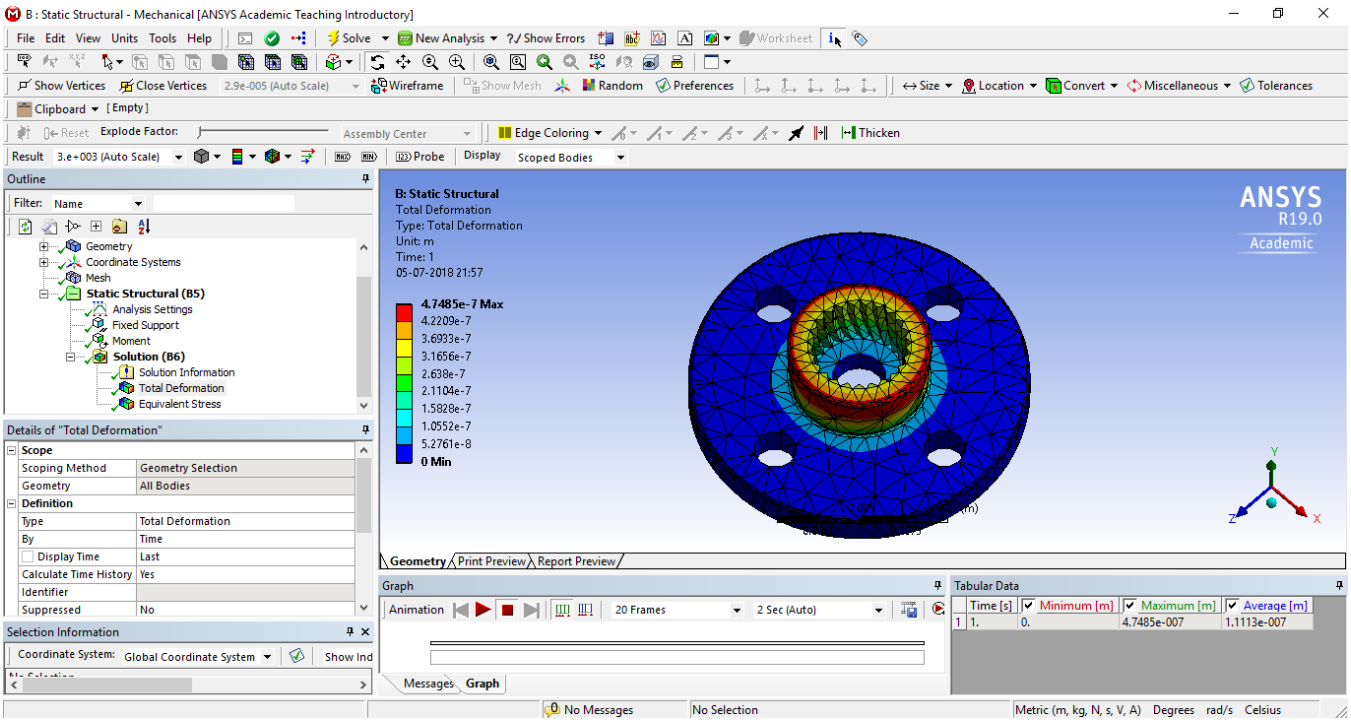
First of all, the geometry of elements is prepared using solid works software which is used to establish three-dimensional model. Then it is imported from solid works to ANSYS workbench for FE analysis. Through the software interface the data exchange of the model is imported to the ANSYS. Structural steel is taken as the material of robot body.

Loads are applied to the elements and the mesh size is taken as default due to student version. The structure is tested to a static analysis in order to obtain total deformation and equivalent stress.

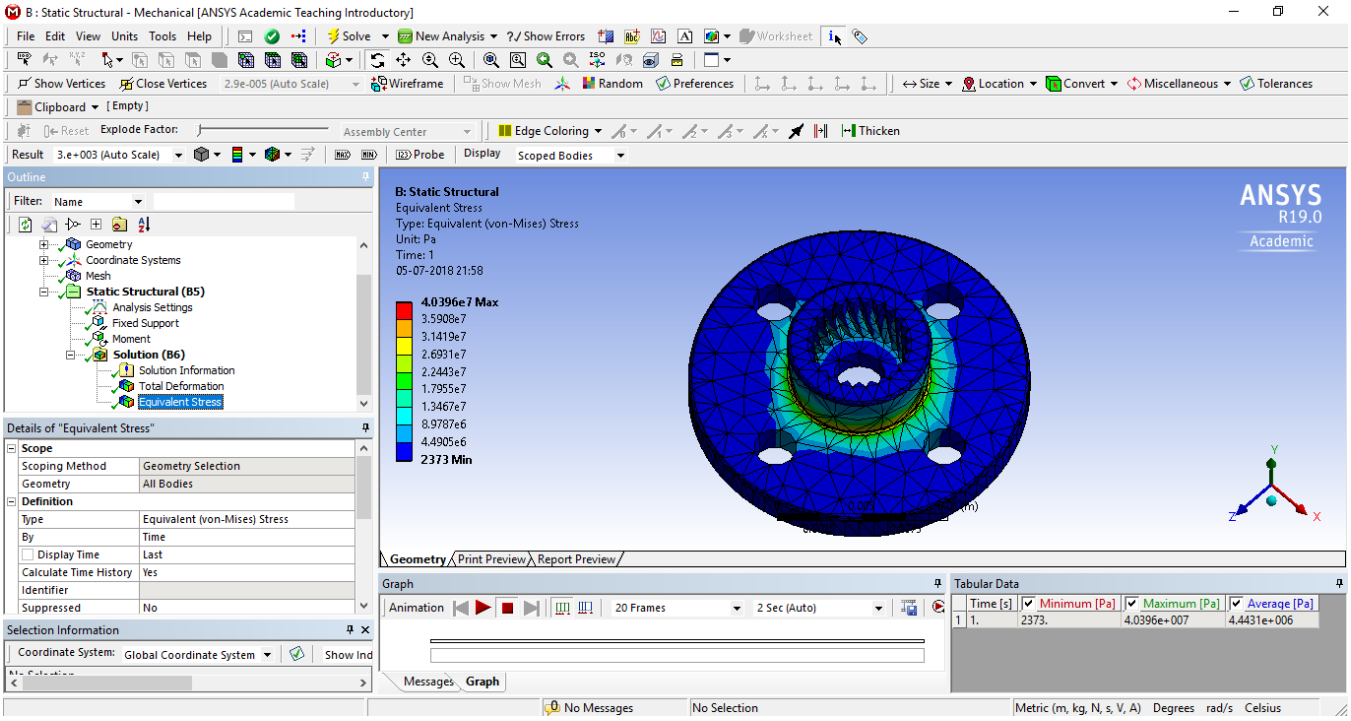
Total deformation and equivalent stresses of different elements are shown in below figures.The dark blue colour indicate the lowest value ofdeformation, light blue colour shows lower value of deformatin, yellow colour indicate higer value of deformation and red colour shows the highest value of deformation which is shown in the left side of below pictures.

Gripper rotator:

Total deformation

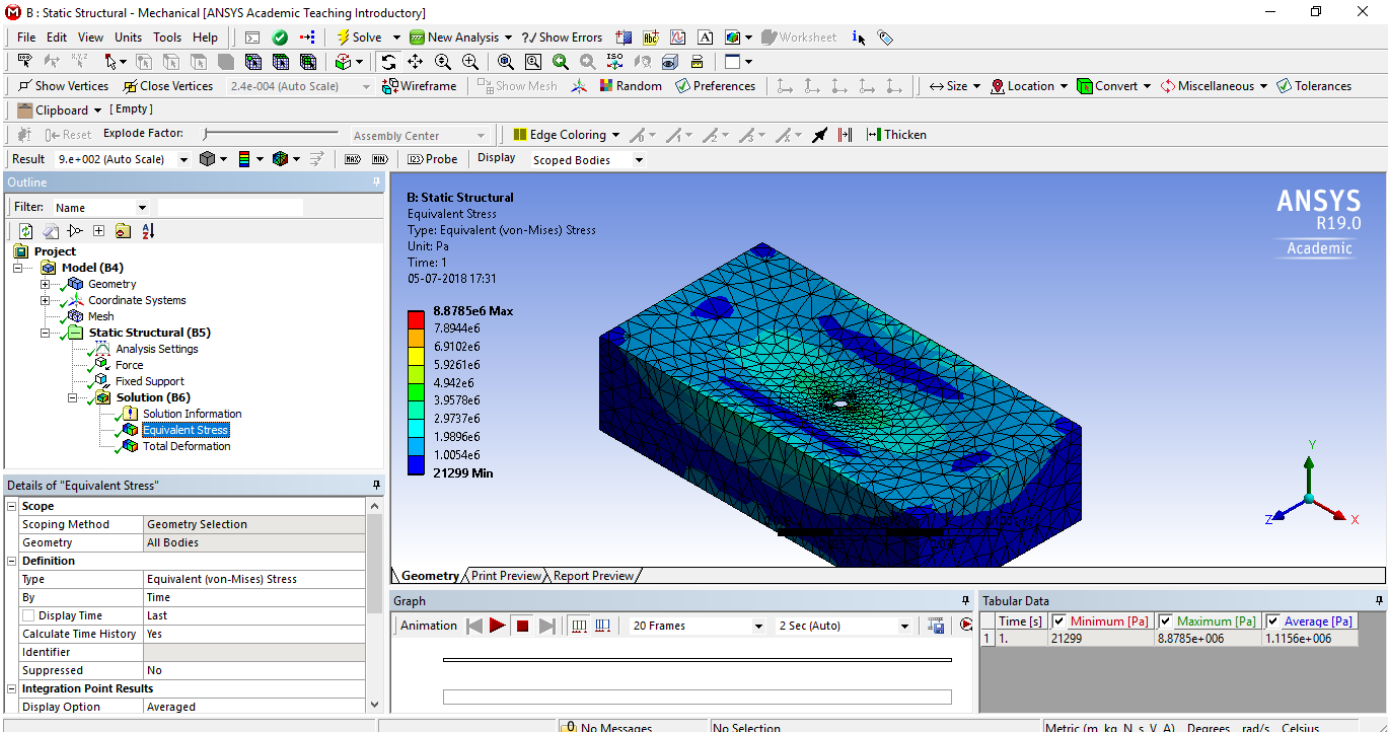


Stress distribution

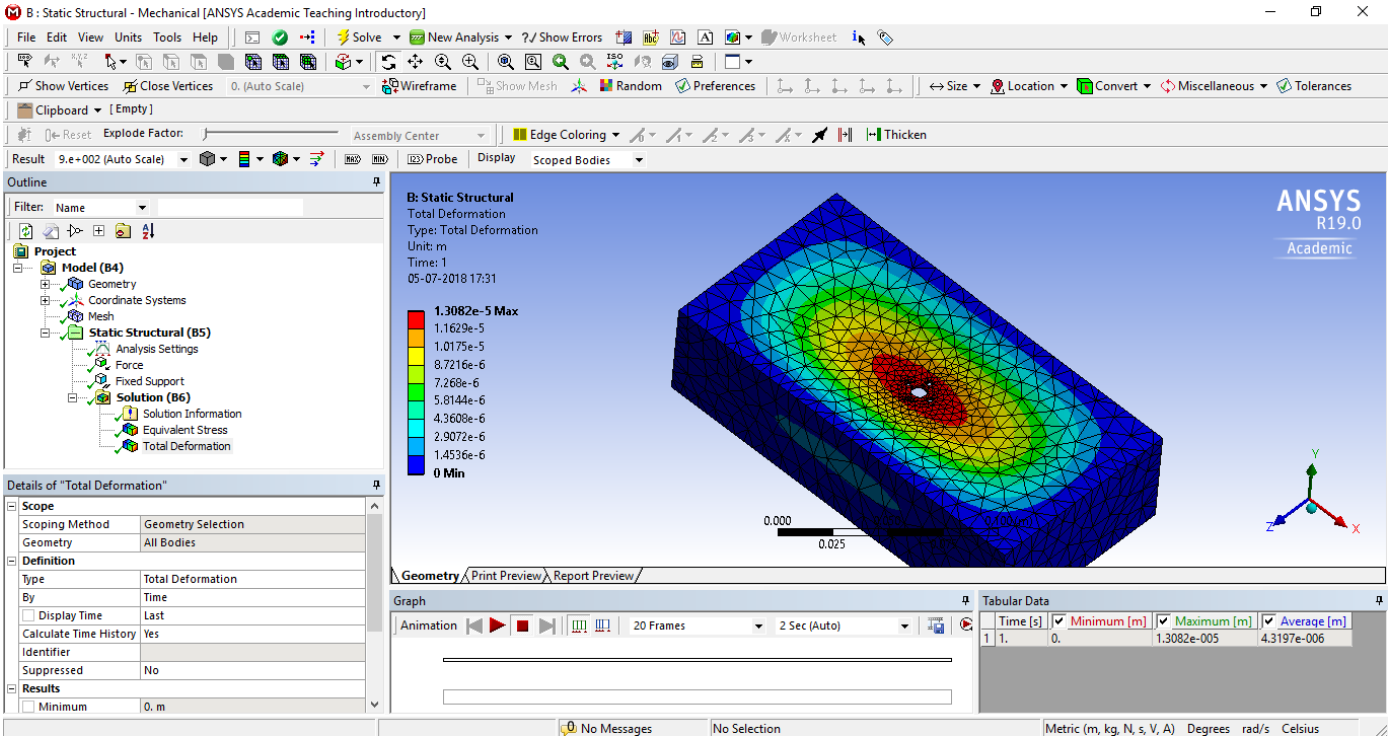


Base of robotic arm:

Stress distribution

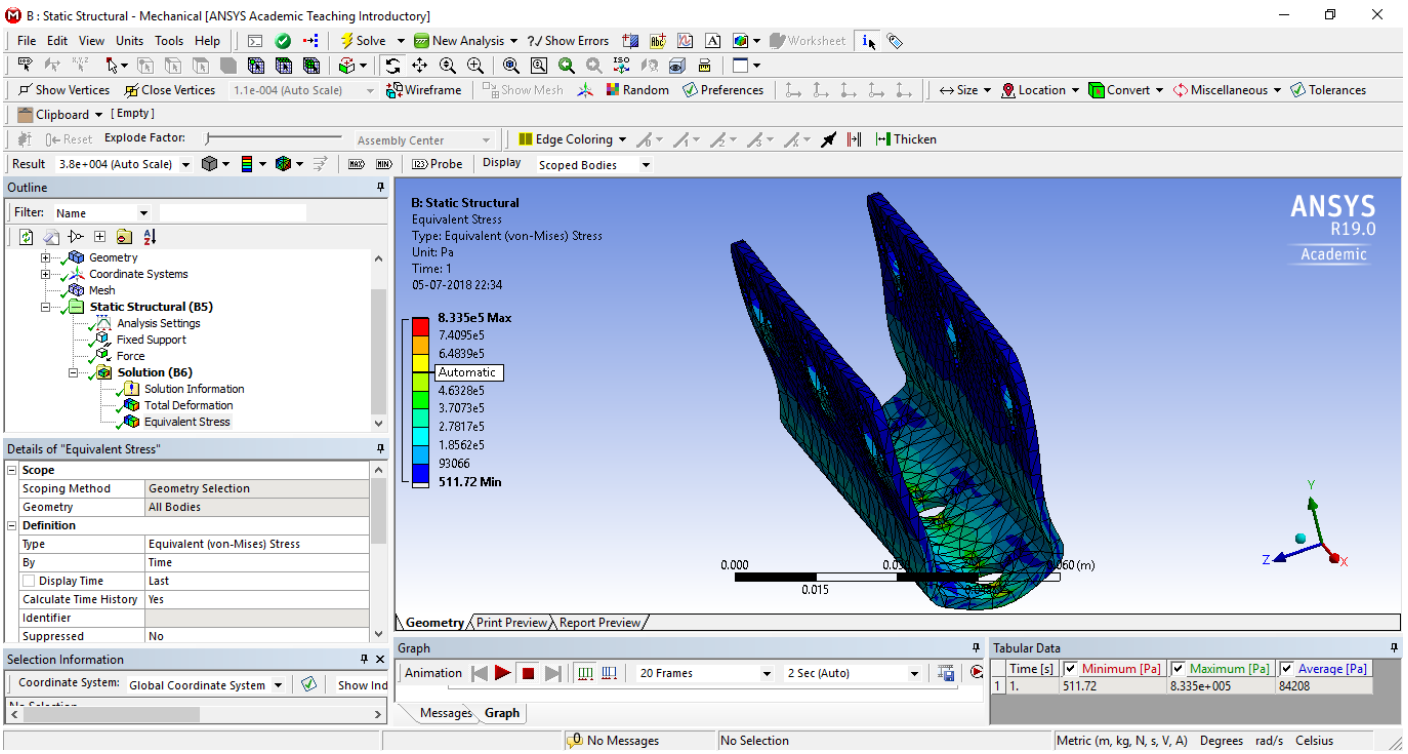


Total deformation

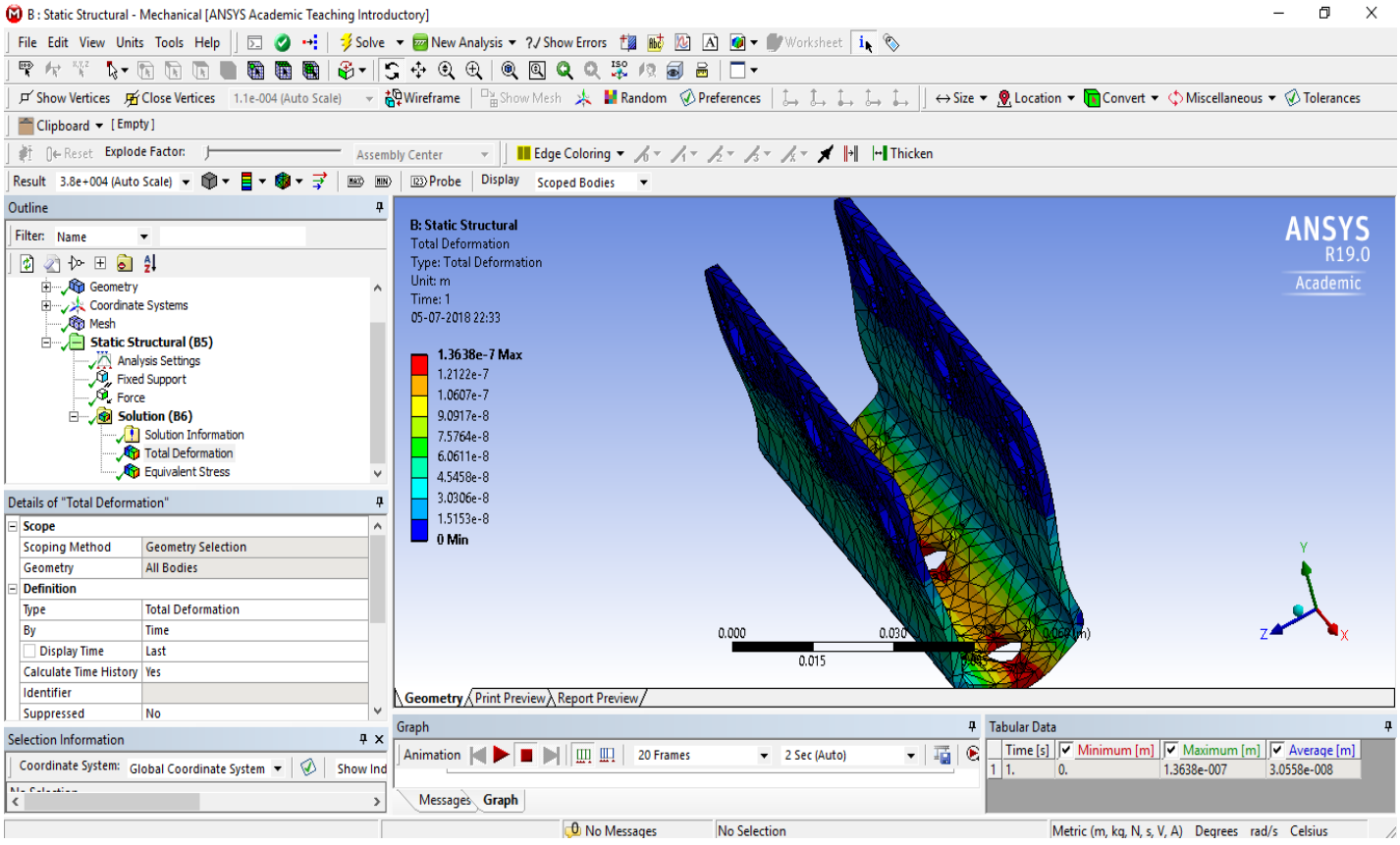


Servo holders:

Equivalent stress

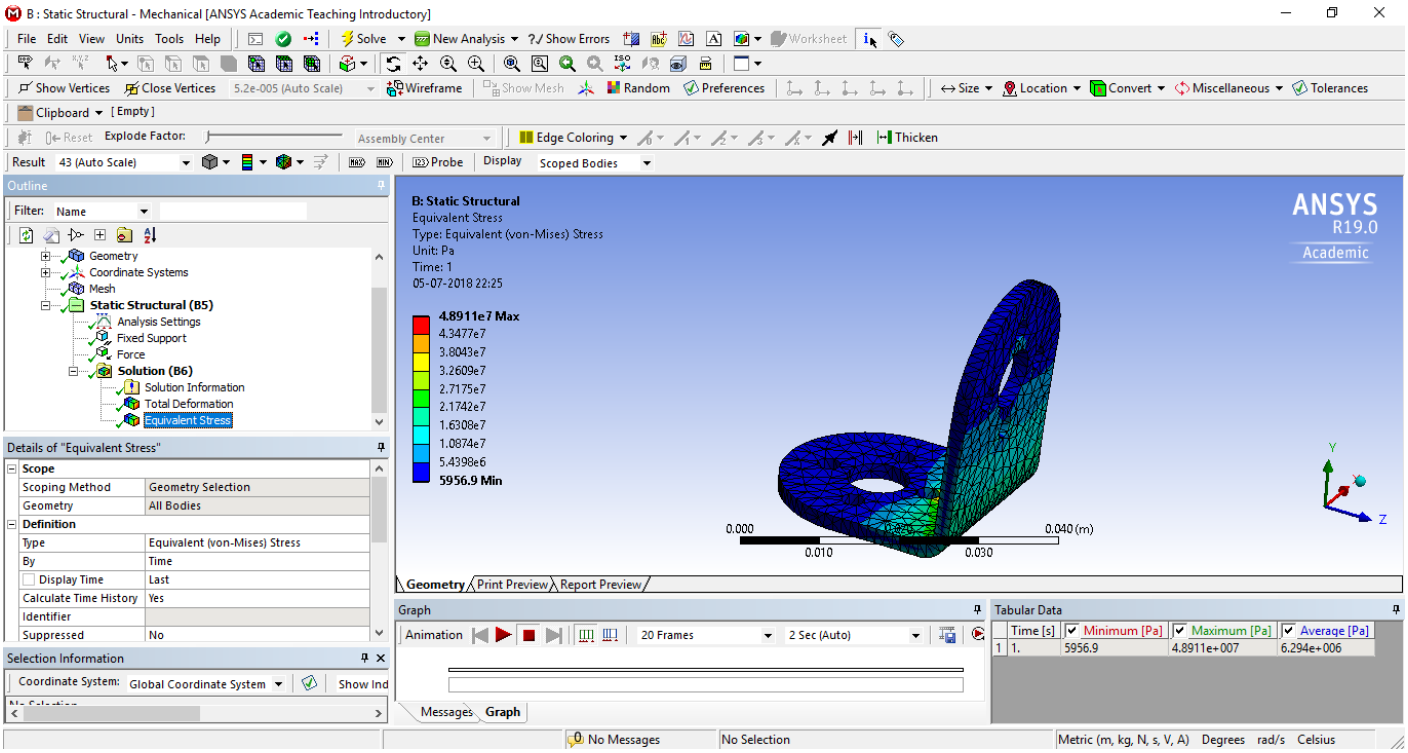


Total deformation

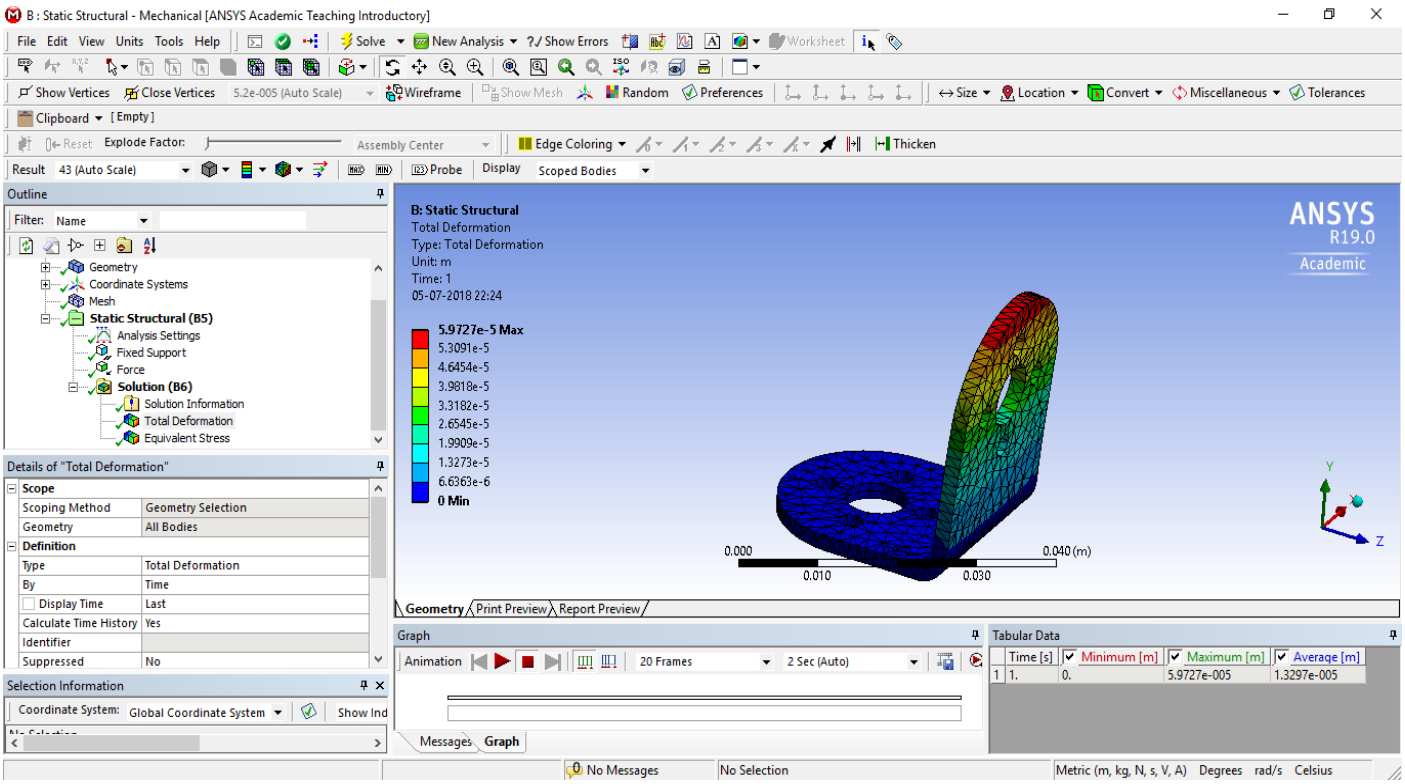


Servo holder:

Equivalent stress

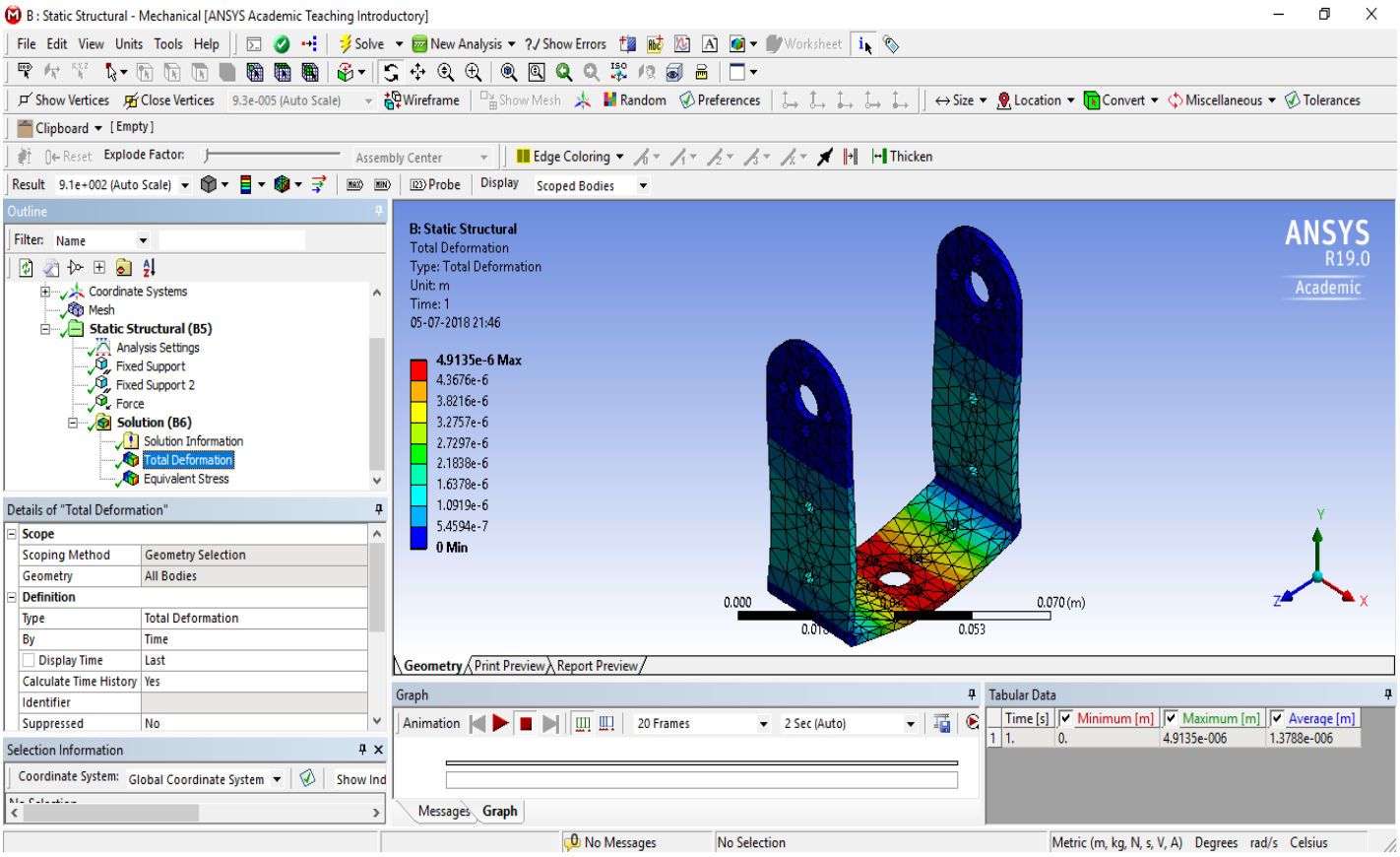


Total deformation



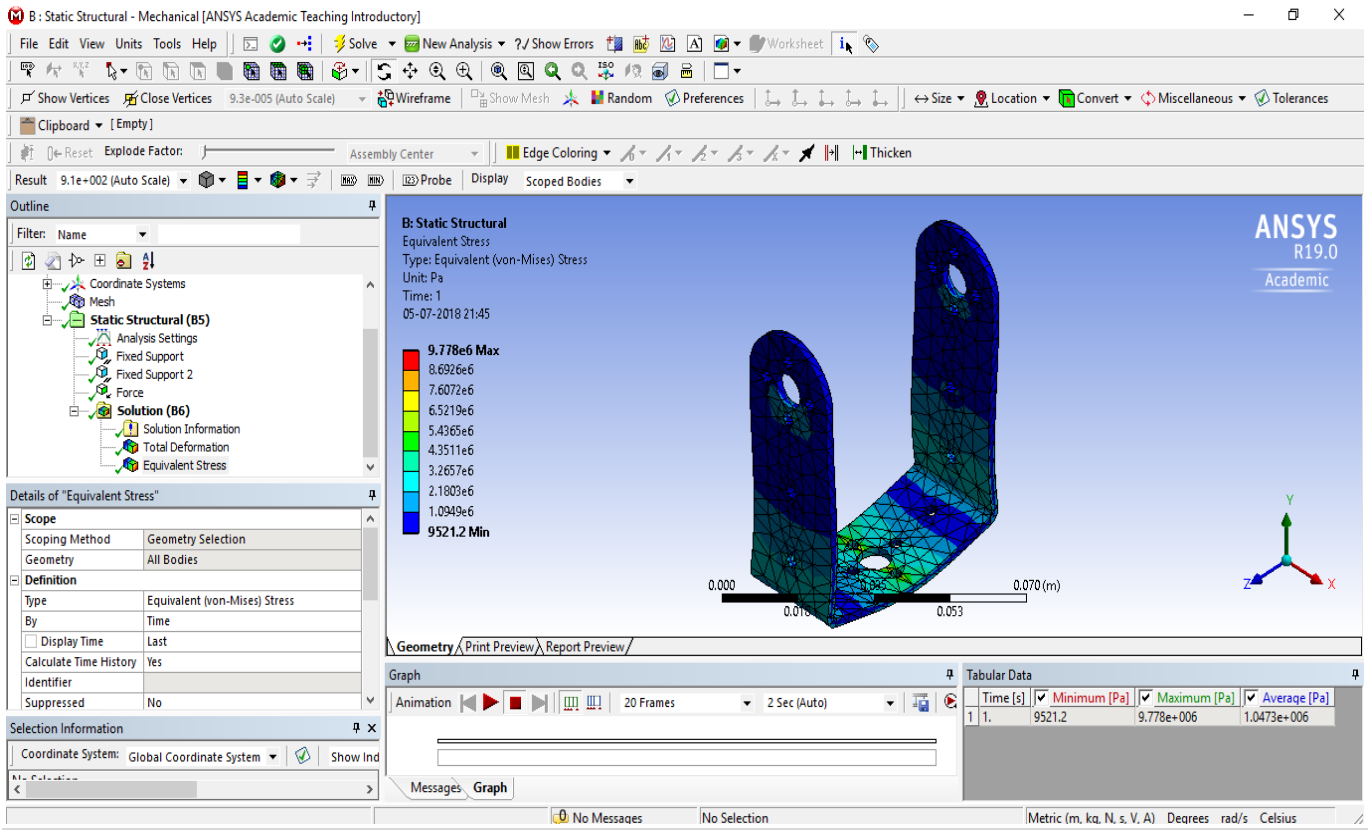
Arm holder:

Total deformation



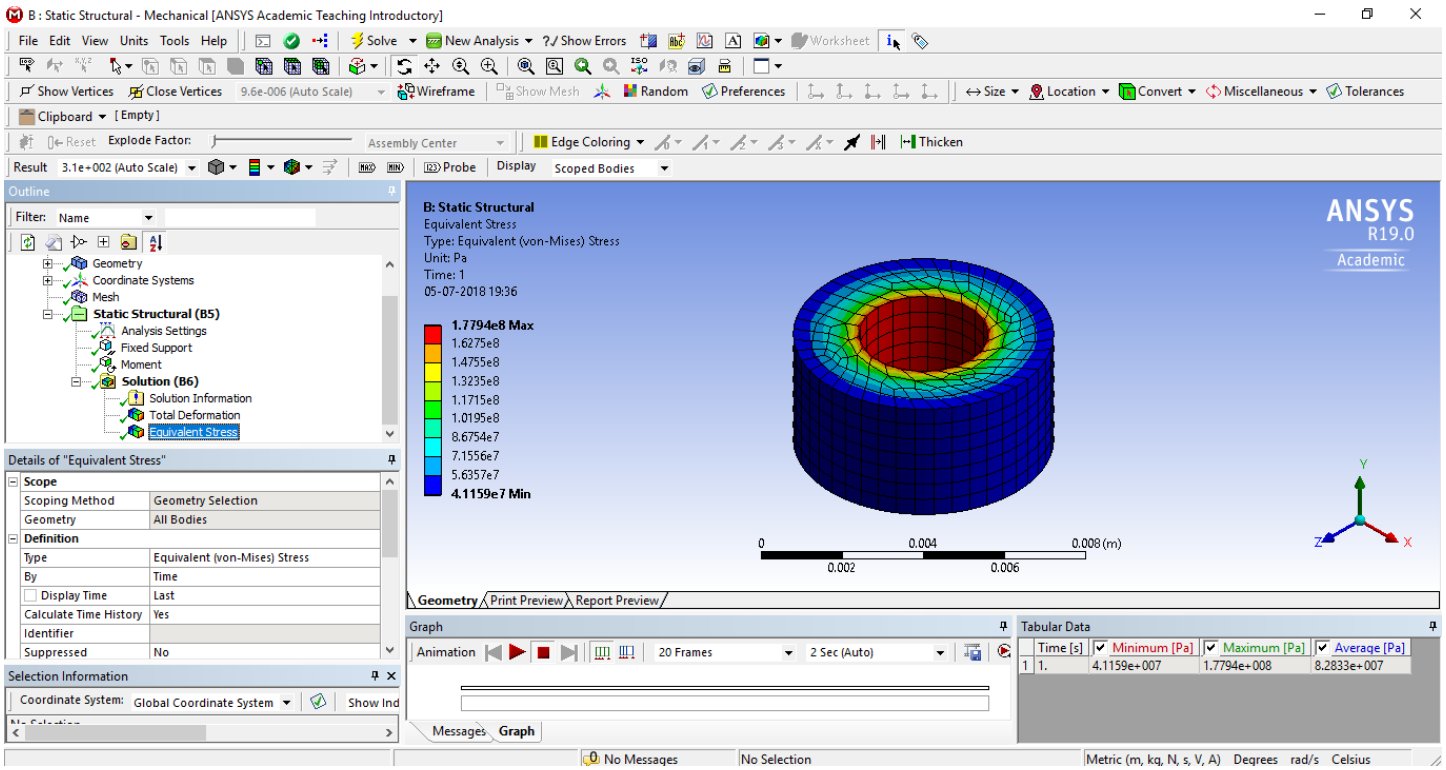
Stress

deformation

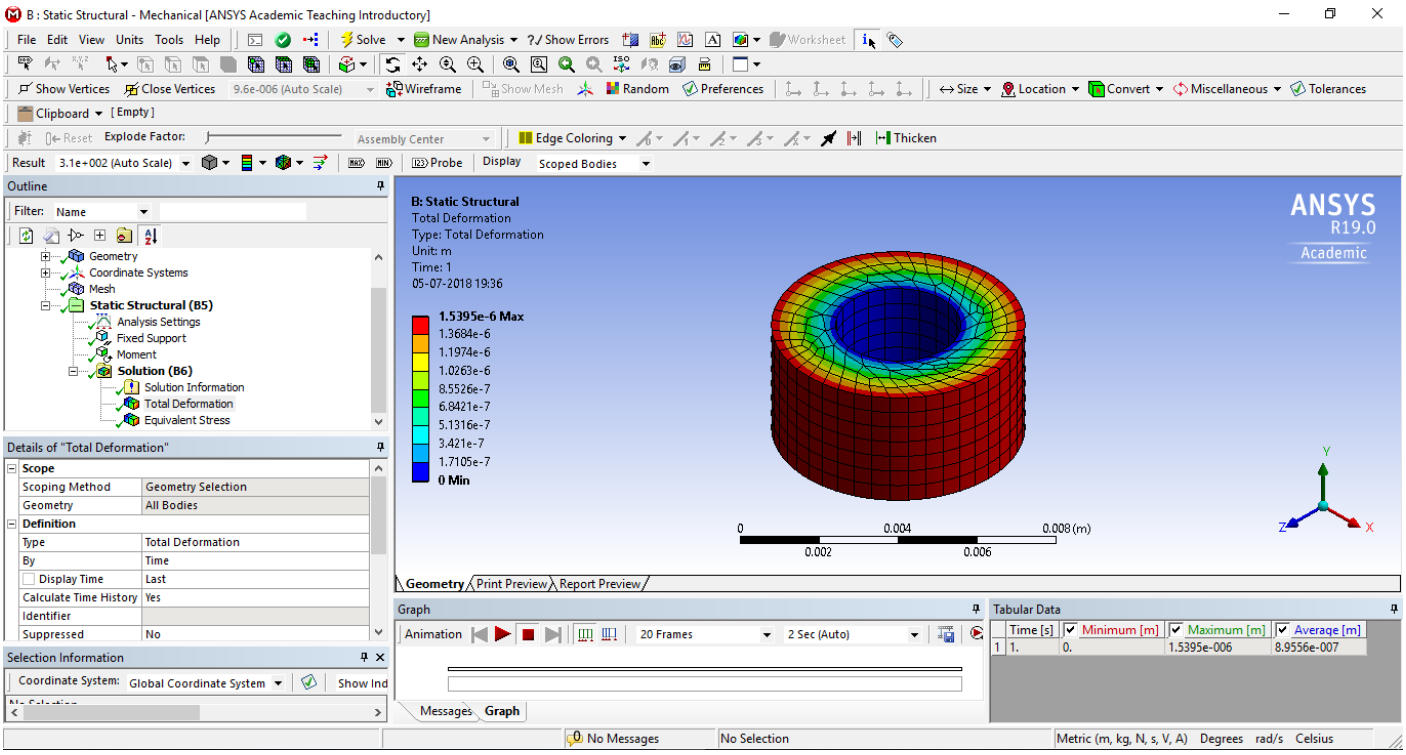


Joint holder:

Stress distribution

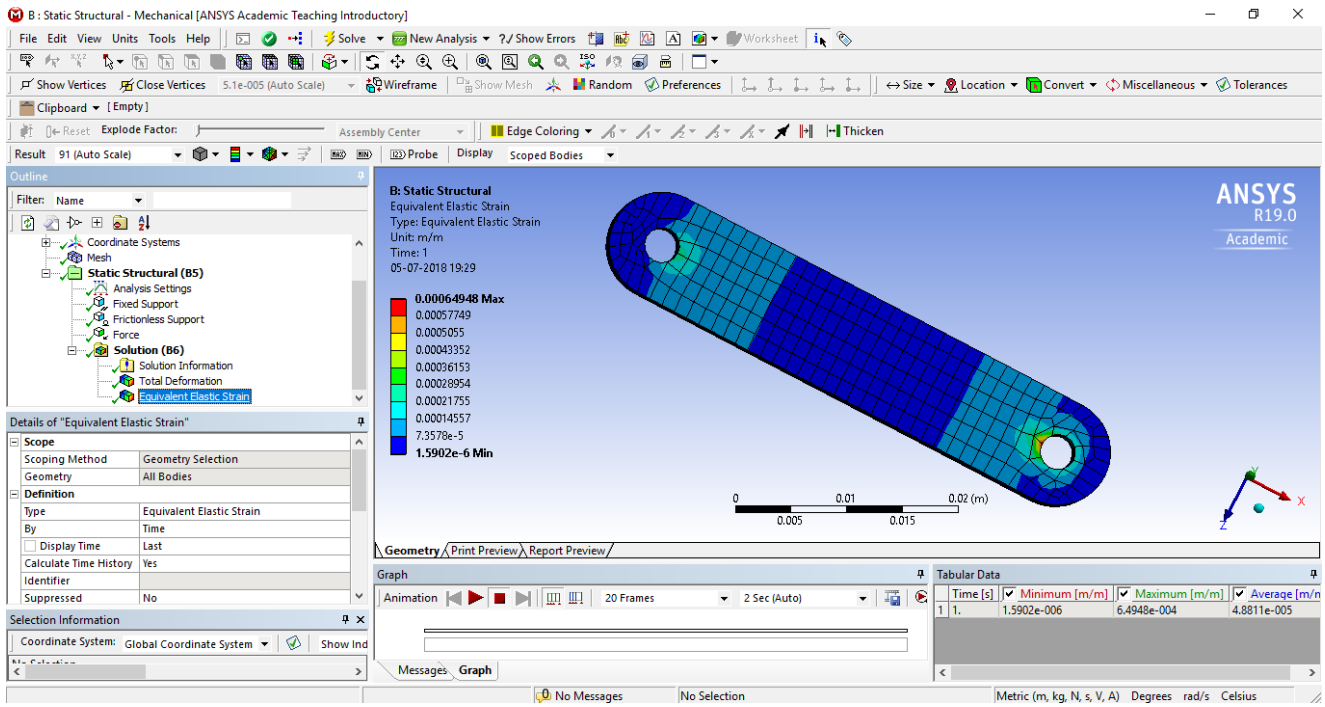


Total deformation

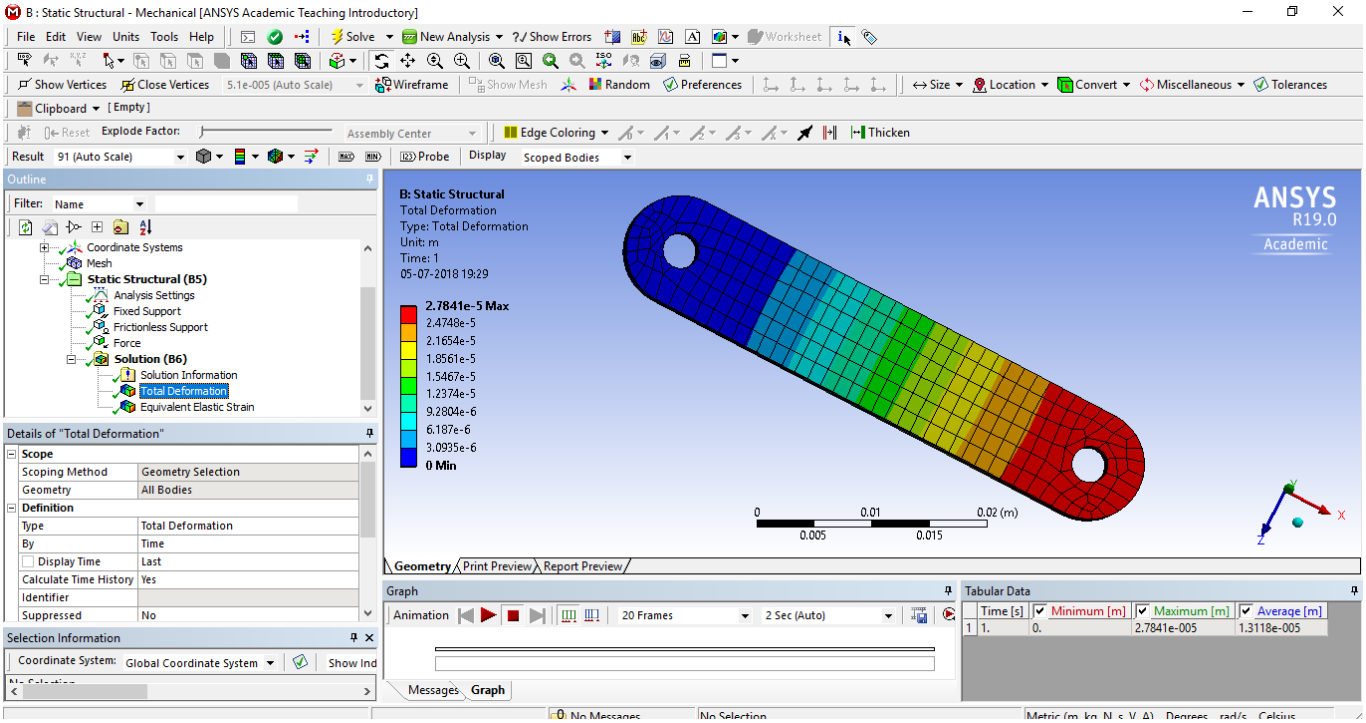


Connecting link:

Stress distribution

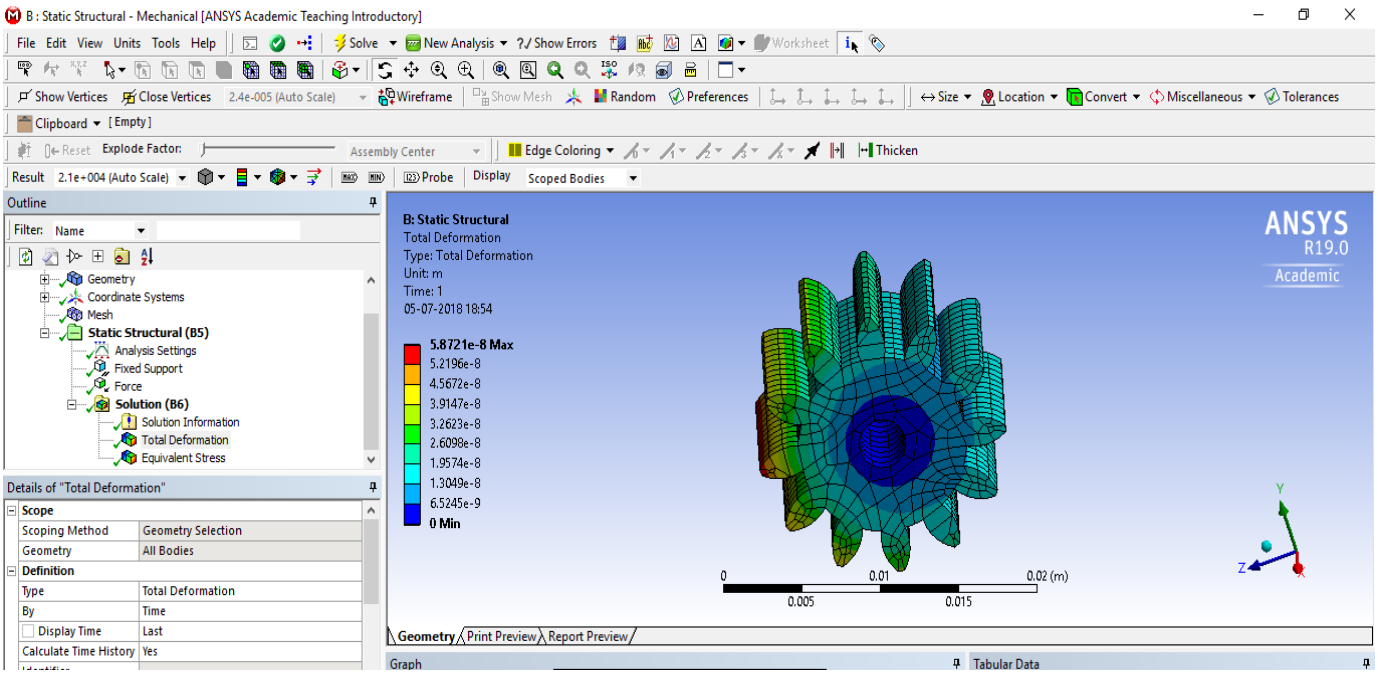


Total deformation

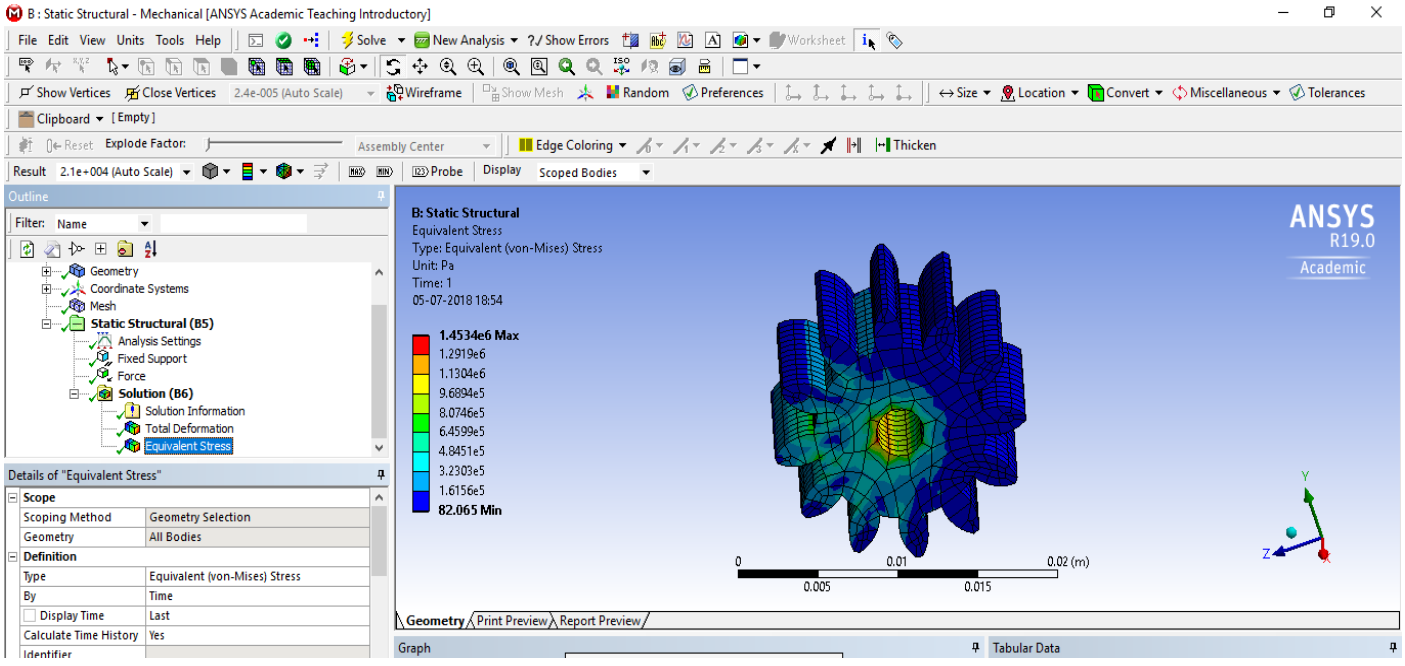


Servo gear:

Total Deformation



Stress distribution



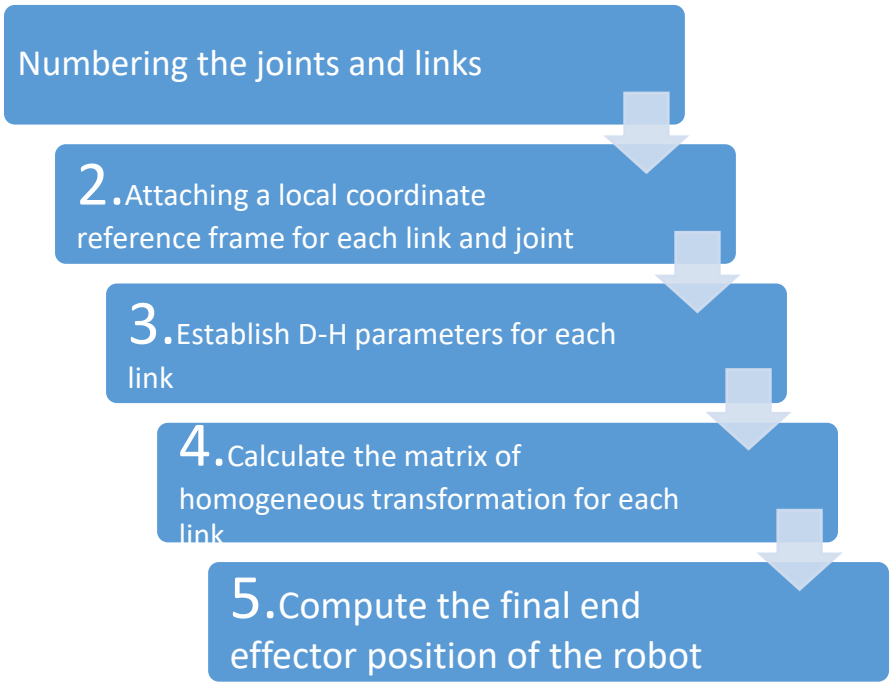
From the analysis, we can conclude that the design is safe and less prone to fatigue.

Kinematic analysis of a robotic arm:

Forward kinematic analysis:

Forward kinematic analysis is simply defined as finding end effector position and orientation from the joint angles i.e. one of the D-H parameters. There are many methods to solve forward but in this I am solving using D-H parameters method.

5 STEP ALGORITHM FOR KINEMATIC ANALYSIS:



Joint-link numbering:

The robotic base and its connection to the first joint are named as link 0. The first joint in the sequence is joint 1. Link 0 is the input link for joint 1, while the output link from joint 1 is link 1 which leads to joint 2. Thus link 1 is, simultaneously, the output link for joint 1 and the input link for joint 2. This joint-link-numbering scheme is followed for all joints and links.
General joint variable is q_i

Rules for assigning co-ordinate frames:
The DH convention is used in the project and the six DOF robot follows the DH rules mentioned for assigning the frames

- **Rule 1**: Z_{i-1} is the axis of revolution of a revolution joint (or) translation of a prismatic joint of joint i.
- **Rule 2**: Axis X_i is perpendicular to Z_{i-1}.
- **Rule 3**: Axis Y_i is derived from X_i and Z_i.

D-H parameters:
In general, there are three parameters for rotation and three parameters for displacement but DH convention converts six parameters to four parameters (three parameters for rotation and one for displacement).They are,

- a_i denotes the link length of link 'i'
- αi denotes the link twist of link 'i'
- θ_i denotes the joint angle of joint 'i', revolute variable.
- d_i denotes the link offset, prismatic variable

Description:

- d - the distance between the previous x-axis and the current x-axis, along the previous z-axis.
- θ - the angle around the z-axis between the previous x-axis and current x-axis.
- a - the length of the common normal, which is the distance between the previous z-axis and the current z-axis
- α - the angle around the common normal to between the previous z-axis and current z-axis.

Equations:
The kinematics equations for the series chain of a robot are obtained using two transformations ([Z] & [X])
[Z] - to characterize the relative movement allowed at each joint
[X] - to define the dimensions of each link

$$[T] = [Z_1][X_1][Z_2][X_2]\dots[X_{n-1}][Z_n],$$

where, [T] is the transformation locating the end-effector. These equations are known as the kinematics equations of the serial chain.

$$[Z_i] = \mathrm{Trans}_{Z_i}(d_i)\,\mathrm{Rot}_{Z_i}(\theta_i), \qquad [X_i] = \mathrm{Trans}_{X_i}(a_{i,i+1})\,\mathrm{Rot}_{X_i}(\alpha_{i,i+1}).$$

where, θ_i, d_i, $\alpha_{i,i+1}$ and $a_{i,i+1}$ are known as the Denavit-Hartenberg parameters.

The matrices associated with these operations are:

$$\mathrm{Trans}_{Z_i}(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{Rot}_{Z_i}(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similarly,

$$\mathrm{Trans}_{X_i}(a_{i,i+1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i,i+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{Rot}_{X_i}(\alpha_{i,i+1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i,i+1} & -\sin\alpha_{i,i+1} & 0 \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since,

$${}^{i-1}T_i = [Z_i][X_i] = \mathrm{Trans}_{Z_i}(d_i)\,\mathrm{Rot}_{Z_i}(\theta_i)\,\mathrm{Trans}_{X_i}(a_{i,i+1})\,\mathrm{Rot}_{X_i}(\alpha_{i,i+1}),$$

Therefore,

$${}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_{i,i+1} & \sin\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i,i+1} & -\cos\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\sin\theta_i \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

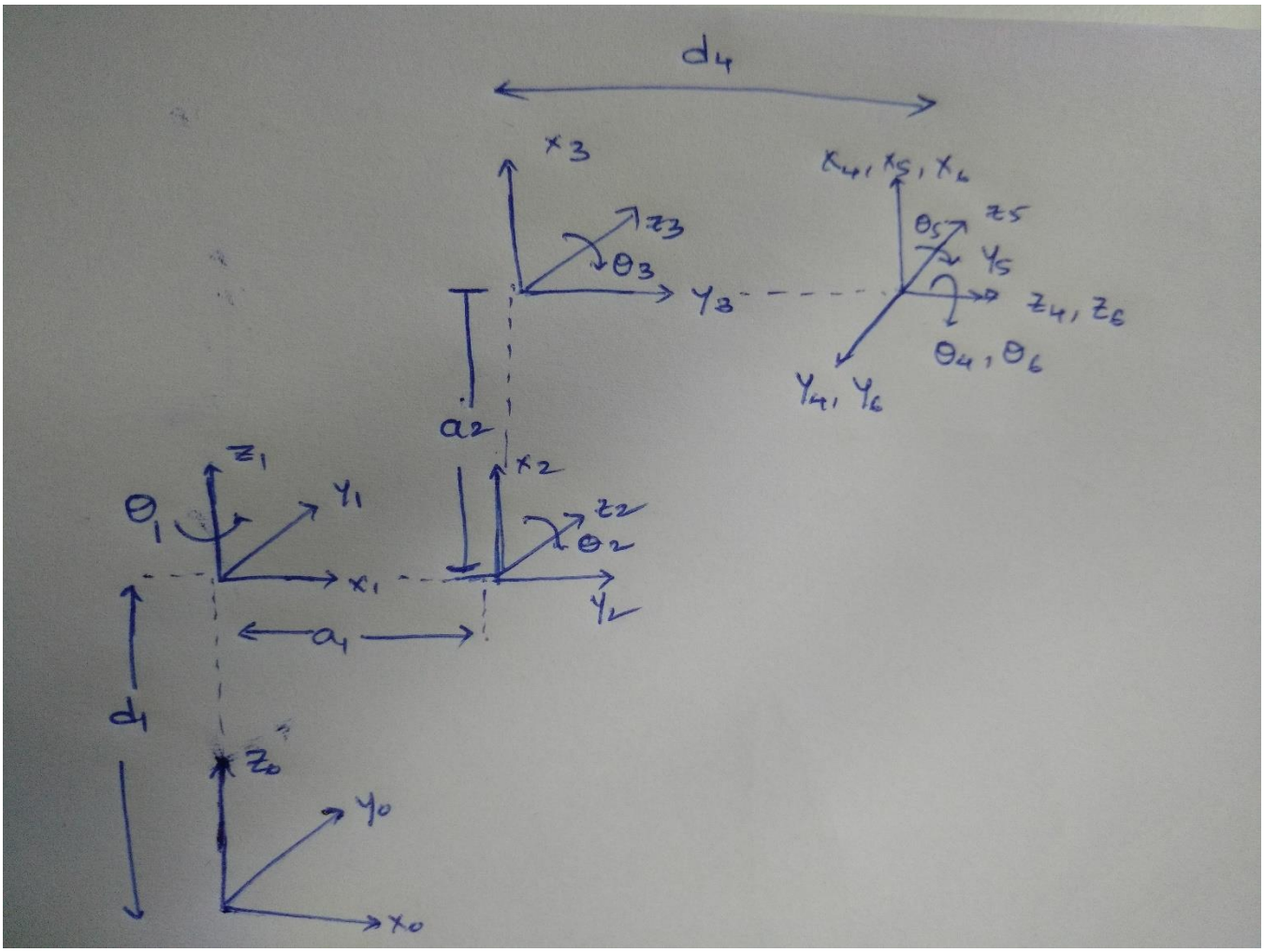
, is known as the Denavit-Hartenberg matrix.

Finally,

$$[T] = {}^0T_n = \prod_{i=1}^{n} {}^{i-1}T_i(\theta_i)$$

, where [T]is the transformation locating the end link.

DH parameters axis diagram:



D-H parameters for robotic arm:

Axis	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	352	θ_1
2	-90	70	0	θ_2-90
3	0	360	0	θ_3
4	-90	0	380	θ_4
5	90	0	0	θ_5
6	-90	0	0	θ_6

assumed that s2 = sin (θ2-90), c2 = cos (θ2-90)

$$
{}^{0}_{1}T = \begin{pmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & a_0 \\ s_{\theta_1}c_{\alpha_0} & c_{\theta_1}c_{\alpha_{01}} & -s_{\alpha_0} & -d_1 s_{\alpha_0} \\ s_{\theta_1}s_{\alpha_0} & c_{\theta_1}s_{\alpha_0} & c_{\alpha_0} & d_1 c_{\alpha_0} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{0}_{1}T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{1}_{2}T = \begin{pmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & a_1 \\ s_{\theta_2}c_{\alpha_1} & c_{\theta_2}c_{\alpha_1} & -s_{\alpha_1} & -d_2 s_{\alpha_1} \\ s_{\theta_2}s_{\alpha_1} & c_{\theta_2}s_{\alpha_1} & c_{\alpha_1} & d_2 c_{\alpha_1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{01}_{2}T = \begin{pmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{2}_{3}T = \begin{pmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & a_2 \\ s_{\theta_3}c_{\alpha_2} & c_{\theta_3}c_{\alpha_2} & -s_{\alpha_2} & -d_3 s_{\alpha_2} \\ s_{\theta_3}s_{\alpha_2} & c_{\theta_3}s_{\alpha_2} & c_{\alpha_2} & d_3 c_{\alpha_2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{2}_{3}T = \begin{pmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{3}_{4}T = \begin{pmatrix} c_{\theta_4} & -s_{\theta_4} & 0 & a_3 \\ s_{\theta_4}c_{\alpha_3} & c_{\theta_4}c_{\alpha_3} & -s_{\alpha_3} & -d_4 s_{\alpha_3} \\ s_{\theta_4}s_{\alpha_3} & c_{\theta_4}s_{\alpha_3} & c_{\alpha_3} & d_4 c_{\alpha_3} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{3}_{4}T = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{4}_{5}T = \begin{pmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & a_4 \\ s_{\theta_5}c_{\alpha_4} & c_{\theta_5}c_{\alpha_4} & -s_{\alpha_4} & -d_5 s_{\alpha_4} \\ s_{\theta_5}s_{\alpha_4} & c_{\theta_5}s_{\alpha_4} & c_{\alpha_4} & d_5 c_{\alpha_4} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{4}_{5}T = \begin{pmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
{}^{5}_{6}T = \begin{pmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & a_5 \\ s_{\theta_6}c_{\alpha_5} & c_{\theta_6}c_{\alpha_5} & -s_{\alpha_5} & -d_6 s_{\alpha_5} \\ s_{\theta_6}s_{\alpha_5} & c_{\theta_5}s_{\alpha_5} & c_{\alpha_5} & d_6 c_{\alpha_5} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{5}_{6}T = \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Once the homogeneous transformation matrix of each link is obtained, forward kinematic chain can be applied to achieve the position and orientation of the robot end-effector with respect to the global reference frame (robot base).

$${}^{0}_{2}T = {}^{0}_{1}T \text{ X } {}^{1}_{2}T$$

$${}^{0}_{2}T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{X} \begin{pmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_1c_2 & -c_1s_2 & -s_1 & c_1a_1 \\ s_1c_2 & -s_1s_2 & c_1 & s_1a_1 \\ -s_2 & -c_2 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}_{3}T = {}^{0}_{2}T \text{ X } {}^{2}_{3}T$$

$${}^{0}_{3}T = \begin{pmatrix} c_1c_2 & -c_1s_2 & -s_1 & c_1a_1 \\ s_1c_2 & -s_1s_2 & c_1 & s_1a_1 \\ -s_2 & -c_2 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}_{3}T = \begin{pmatrix} c_1c_2c_3 - c_1s_2s_3 & -(c_1c_2s_3 + c_1s_2c_3) & -s_1 & c_1c_2a_2 + c_1a_1) \\ s_1c_2c_3 - s_1s_2s_3 & -(s_1c_2s_3 + s_1s_2c_3) & c_1 & s_1c_2a_2 + s_1a_1) \\ -(s_2c_3 + c_2s_3) & s_2s_3 - c_2c_3 & 0 & -s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}_{3}T = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(c_2a_2 + a_1) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(c_2a_2 + a_1) \\ -s_{23} & -c_{23} & 0 & -s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{4}_{6}T = {}^{4}_{5}T \text{ X } {}^{5}_{6}T$$

$${}^{4}_{6}T = \begin{pmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_5c_6 & -c_5s_6 & -s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{3}_{6}T = {}^{3}_{4}T \text{ X } {}^{4}_{6}T$$

$${}^{3}_{6}T = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_5c_6 & -c_5s_6 & -s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{3}_{6}T = \begin{pmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}_{6}T = {}^{0}_{3}T \text{ X } {}^{3}_{6}T$$

$${}^{0}_{6}T = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(c_2a_2 + a_1) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(c_2a_2 + a_1) \\ -s_{23} & -c_{23} & 0 & -s_2a_2 + d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}_{6}T = \begin{pmatrix} r11 & r12 & r13 & x \\ r21 & r22 & r23 & y \\ r31 & r32 & r33 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where,

$r_{11} = c_1c_{23}(c_4c_5c_6 - s_4s_6) - c_1s_{23}s_5c_6 + s_1(s_4c_5c_6 + c_4s_6)$
$r_{12} = c_1c_{23}(-c_4c_5s_6 - s_4c_6) + c_1s_{23}s_5s_6 - s_1(s_4c_5s_6 - c_4c_6)$
$r_{13} = -c_1c_{23}c_4s_5 - c_1s_{23}c_5 - s_1s_4s_5$
$r_{21} = s_1c_{23}(c_4c_5c_6 - s_4s_6) - s_1s_{23}s_5c_6 - c_1(s_4c_5c_6 + c_4s_6)$
$r_{22} = s_1c_{23}(-c_4c_5s_6 - s_4c_6) + s_1s_{23}s_5s_6 + c_1(s_4c_5s_6 - c_4c_6)$
$r_{23} = -s_1c_{23}c_4s_5 - s_1s_{23}c_5 + c_1s_4s_5$
$r_{31} = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6$
$r_{32} = -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6$
$r_{33} = s_{23}c_4s_5 - c_{23}c_5$
$x = -d_4c_1s_{23} + c_1(c_2a_2 + a_1)$
$y = -d_4s_1s_{23} + s_1(c_2a_2 + a_1)$
$z = -s_2a_2 + d_1 - d_4c_{23}$

Now, it is also possible to find the position of the tip (TCP) with respect to the robot base. According to the robot frame assignment, it is simply a transition along the z axis of frame {6} by d6 (65 mm).
Therefore, the final position of the end effector with respect to the robot global reference frame can be expressed as:

$Ptip = {}^0_6T \; X \; P^6$

$$Ptip = \begin{pmatrix} r11 & r12 & r13 & x \\ r21 & r22 & r23 & y \\ r31 & r32 & r33 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} 0 \\ 0 \\ d6 \\ 1 \end{pmatrix} = \begin{pmatrix} d6\,X\,r13 + x \\ d6\,X\,r23 + y \\ d6\,X\,r33 + z \\ 1 \end{pmatrix}$$

Matlab code for finding end effector position:

Execute | Embed demo.m STDIN

```
% MATLAB CODE FOR FORWAR KINEMETIC ANALYSIS
%NON RETURN FUNCTION OF THE MAIN PROGRAM TO COMBINE ALL THE FUNCTIONS
%TOGETHER IN ONE SCRIPT
function [ NONRETURNFN ] = FORWARD( )
% DECLARING DH PARAMETERS
a0 = 0; d1 = 352; alpha0 = 0;
a1 = 70; d2 = 0; alpha1 = -pi/2;
a2 = 360; d3 = 0; alpha2 = 0;
a3 = 0; d4 = 380; alpha3 = -pi/2;
a4 = 0; d5 = 0; alpha4 = pi/2;
a5 = 0; d6 = 0; alpha5 = -pi/2;
% USER INTERFACE
theta1 = input ('ENTER THE VALUE OF THETA1 IN DEGREE = ');
theta2 = input ('ENTER THE VALUE OF THETA2 IN DEGREE = ');
theta3 = input ('ENTER THE VALUE OF THETA3 IN DEGREE = ');
theta4 = input ('ENTER THE VALUE OF THETA4 IN DEGREE = ');
theta5 = input ('ENTER THE VALUE OF THETA5 IN DEGREE = ');
theta6 = input ('ENTER THE VALUE OF THETA6 IN DEGREE = ');
% CALL THE DH FUNCTION TO CALCULATE THE HOMOGENOUS TRANSFORMATION MATRICES
T10 = DHFUNCTION(a0,alpha0,d1,theta1*pi/180)
T21 = DHFUNCTION(a1,alpha1,d2,(theta2-90)*pi/180)
T32 = DHFUNCTION(a2,alpha2,d3,theta3*pi/180)
T43 = DHFUNCTION(a3,alpha3,d4,theta4*pi/180)
T54 = DHFUNCTION(a4,alpha4,d5,theta5*pi/180)
T65 = DHFUNCTION(a5,alpha5,d6,theta6*pi/180)
T20 = T10*T21;
T30 = T20*T32;
T64 = T54*T65;
T63 = T43*T64;
T60 = T30*T63
% THE POSTION OF THE END EFEECTOR AT JOINT 6
Xw = T60(1,4);
Yw = T60(2,4);
Zw = T60(3,4);
P6 = [Xw;Yw;Zw]
% THE POSTION OF THE END EFEECTOR AT THE TCP
PTCP= T60*[0;0;65;1]
% Modifed DH TRANSFORM FUNCTION
function T = DHFUNCTION(ai,alphai,di,thetai)
T = [ cos(thetai),              -1.*sin(thetai),              0,              ai ;
    sin(thetai).*cos(alphai),   cos(thetai).*cos(alphai),   -1.*sin(alphai),   1*di.*sin(alphai);
    sin(thetai).*sin(alphai),   cos(thetai).*sin(alphai),    cos(alphai),      di.*cos(alphai);
    0,                                0,                    0,                1             ];
end
end
```

Inverse Kinematics:

Inverse kinematics is used to calculate the joint angles required to achieve the desired position and orientation in the robot workspace. In broad, there are two approaches of solution, the analytical and geometrical approaches. Since three consecutive axes of the robot intersect at a common point, Pieper's solution can be applied. Pieper's approach works on the principle of separating the position into two, one solution for Θ1, Θ2 and Θ3 from the orientation solution to solve for Θ4, Θ5 and Θ6. Therefore, a geometrical approach is initially implemented to find out the joint variables Θ1, Θ2 and Θ3 that will define the end effector position in space, while an analytical solution is applied to calculate the angles Θ4, Θ5 and Θ6 which will describe the end-effector orientation.

Geometrical solution:

According to the frame assignment shown hand drawn picture, x and y components of frame {1} is the identical as frame {0} because there is only a Z-directional offset between the two frames.

Position	Joint angles	X vector	Y vector	Z vector
0	Θ1 = 0, Θ2 = 0, Θ3 = 0	450	0	712
1	Θ1 = 0, Θ2 = 0, Θ3 = -90	70	0	1092
2	Θ1 = 0, Θ2 = 0, Θ3 = 50	314	0	420.9
3	Θ1 = 0, Θ2 = 110, Θ3 = -90	765	0	98.9
6	Θ1 = 0, Θ2 = -90, Θ3 = 50	1.1	0	596
7	Θ1 = 0, Θ2 = 110, Θ3 =-230	218	0	558
8	Θ1 = 0, Θ2 = -90, Θ3 = -90	-670	0	352

Therefore, the projection of the wrist components on x-y plane of frame {0} has the same components on frame {1}. In addition, since both link two and three are planar, the position vector in y direction varies with respect to θ1 only. Thus, two possible solutions for θ1 can be achieved by simply applying the arctangent function.

$$\theta 1 = \text{atan2}(Py, Px),$$
$$\theta 11 = \Pi + \theta 1.$$

The solutions of θ2 and θ3 are achieved by considering the plane, shown in Figure 5, formed by the second and third planar links with respect to the robot global reference frame.

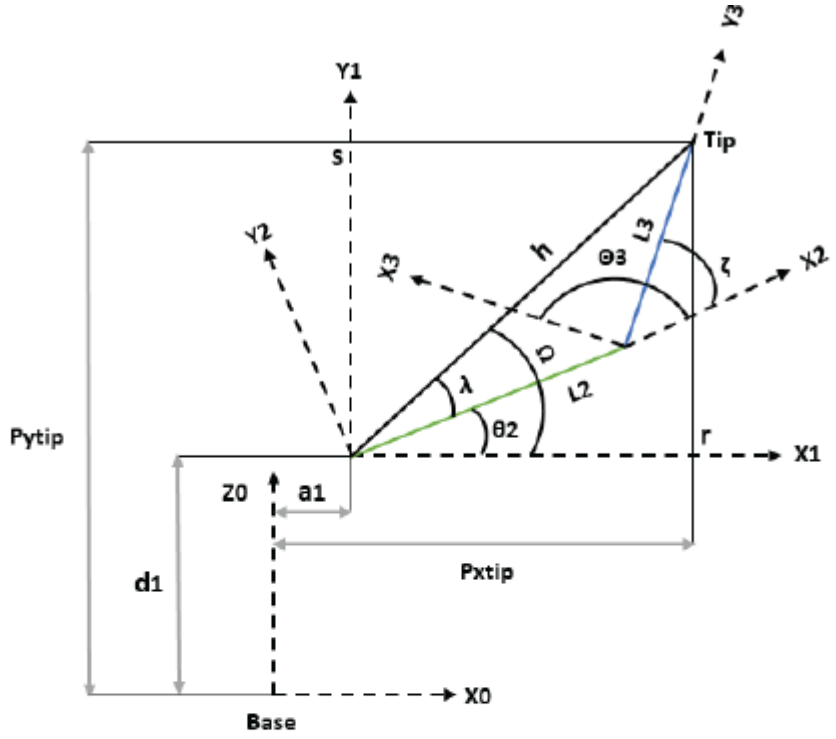


Figure 5. Projection of links two and three onto the x y plane

The cosine low is used to solve for θ3 as follow:

$h^2 = (L_2)^2 + (L_3)^2 - 2 \times L_2 \times L_3 \cos(180 - \zeta)$

Since the position is given with respect to the robot tip (TCP), L_3 should be equal to d4 + d6.
While, $L_2 = a_2$, $h^2 = s^2 + r^2$, $\cos(180 - \zeta) = -\cos(\zeta)$.

$s^2 + r^2 = (a_2)^2 + (d_4 + d_6)^2 + 2 \times a_2 \times (d_4 + d_6) \cos(\zeta)$

$$\text{Cos}(\zeta) = \frac{[s2 + r2 - (a2)2 - (d4 + d6)\,2]]}{2 \times a2 \times (d4 + d6)}$$

Now, we should have the value of (s) and (r) in term of P_{xtip}, P_{ytip}, P_{ztip} and **θ1**.
$S = (P_{ztip} - d1)$

$r = \pm\sqrt{(Pxtip - a1\cos(\theta1))2 + (Pytip - a1\sin(\theta1))2}$, Sub. (s) and (r) in (4.3) yield:

$$\text{Cos}(\zeta) = \frac{[(Pztip - d1)2 + (Pxtip - a1\cos(\theta1))2 + (Pytip - a1\sin(\theta1))2 - (a2)2 - (d4 + d6)\,2]}{2 \times a2 \times (d4 + d6)}$$

$\text{Sin}(\zeta) = \pm\sqrt{1 - \text{Cos}2(\zeta)}$

$\zeta = \text{atan2}(\text{Sin}(\zeta), \text{Cos}(\zeta))$

Finally, $\theta3 = -(90 + \zeta)$

The negative sign in θ3 indicates that the rotation occurred in the opposite direction. Likewise, we can follow the same procedure to solve for θ2 using similar trigonometric relationships.

$\theta 2 = \Omega - \lambda$

$\Omega = \text{atan2}\ (s, r)$

$\lambda = \text{atan2}\ ((d_4+d_6)\ \sin\ (\zeta),\ a_2 + (d_4+d_6)\ \cos\ (\zeta))$

$\theta 2 = \text{atan2}\ (s, r) - \text{atan2}\ [(d_4+d_6)\ \sin\ (\zeta),\ a_2 + (d_4+d_6)\ \cos\ (\zeta)]$, sub the values of (s) and (r) yield:

$\theta 2 = \text{atan2}\ [(P_{ztip} - d1), \pm\sqrt{(Pxtip - a1\ \cos\ (\theta 1))2 + (Pytip - a1\ \sin\ (\theta 1))2}]$

$- \text{atan2}\ [(d_4+d_6)\ x\ \sin\ (\zeta),\ a_2 + (d_4+d_6)\ \cos\ (\zeta)].$

Again the rotation occurred in the opposite direction of the z axis as well as there are an initial rotation of 90^0 between axis 1 and axis 2. Therefore, the final value of θ2 equal to:

$\theta 2 = -((\Omega - \lambda) - 90).$

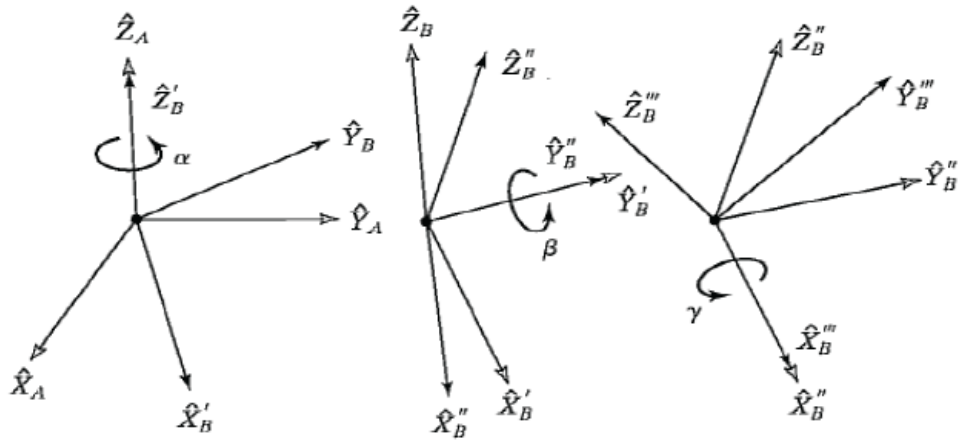
It is important to say that any position within the robot workspace can be achieved with many orientations. Therefore, multiple solutions exist for the variables Θ1, Θ2 and Θ3 due to the nature of trigonometric functions. As noticed above, every solution step resulted in two values that will be used in the next step, and so on. For example, there are four solutions for ζ that resulted from two different values of θ1 (θ1 and θ11), this procedure gives four solutions for θ3, each solution corresponds to different robot configurations of elbow-up and elbow-down representations. These solutions can be listed in Table below to illustrate all the possible solution set.

Solution	THETA1	THETA3	THETA2	Set
1	Θ1	Θ3	Θ2	SET 1
2	Θ1	Θ3	Θ22	
3	Θ1	Θ33	Θ2i	SET 2
4	Θ1	Θ33	Θ22i	
5	Θ11	Θ3i	Θ2j	SET 3
6	Θ11	Θ3i	Θ22j	
7	Θ11	Θ33i	Θ2k	SET 4
8	Θ11	Θ33i	Θ22k	

Analytical solution:

After solving the first inverse kinematic sub-problem which gives the required position of the end effector, the next step of the inverse kinematic solution will deal with the procedure of solving the orientation sub-problem to find the joint angles Θ4, Θ5 and Θ6. This can be done using Z-Y-X Euler's formula. As the orientation of the tool frame with respect to the robot base frame is described in term of Z-Y-X Euler's rotation, this means that each rotation will take place about an axis whose location depends on the previous rotation [3]. The Z-Y-X Euler's rotation is shown below in Figure 6.

The final orientation matrix that results from these three consecutive rotations will be as follow:



$${}^0_6R = R_{z'y'x'} = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma)$$

$${}^0_6R = \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{pmatrix} X \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{pmatrix}$$

$${}^0_6R = \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{pmatrix}$$

Recall the forward kinematic equation,

$${}^0_3R = \begin{pmatrix} c_1 c_{23} & -c_1 s_{23} & -s_1 \\ s_1 c_{23} & -s_1 s_{23} & c_1 \\ -s_{23} & -c_{23} & 0 \end{pmatrix}$$

$${}^3_6R = ({}^0_3R)^T \; {}^0_6R$$

$${}^3_6R = \begin{pmatrix} c_1 c_{23} & s_1 c_{23} & -s_{23} \\ -c_1 s_{23} & -s_1 s_{23} & -c_{23} \\ -s_1 & c_1 & 0 \end{pmatrix} x \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{pmatrix}$$

$${}^3_6R = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix}$$

However, it can be concluded that the last three intersected joints form a set of ZYZ Euler angles with respect to frame {3}. Therefore, these rotations can be expressed as:

$$R_{z'y'z'} = {}^{3}_{6}R = R_z(\alpha)\, R_y(\beta)\, R_z(\gamma)$$

$${}^{3}_{6}R = \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{pmatrix} X \begin{pmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^{3}_{6}R = \begin{pmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{pmatrix}$$

Where ${}^{3}_{6}R$ is given above as

$${}^{3}_{6}R = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix}$$

It is possible now to use the ZYZ Euler's angles formula to obtain the solutions for Θ4, Θ5 and Θ6 where

$$\theta_5 = \beta = atan2\left(+\sqrt{g_{31}^2 + g_{32}^2}, g_{33}\right)$$

$$\theta_4 = \alpha = atan2\left(\frac{g_{32}}{s_\beta}, \frac{-g_{31}}{s_\beta}\right)$$

$$\theta_6 = \gamma = atan2\left(\frac{g_{23}}{s_\beta}, \frac{g_{13}}{s_\beta}\right)$$

For each of the eight solutions achieved from the geometric approach for Θ1, Θ2 and Θ3, there is another flipped solution of Θ4, Θ5 and Θ6 that can be obtained as:

$$\theta_{55} = \beta' = atan2\left(-\sqrt{g_{31}^2 + g_{32}^2}, g_{33}\right), \text{ Or simply } \Theta 55 = -\Theta 5$$

$$\theta_{44} = \alpha = atan2\left(\frac{g_{32}}{s_{\beta'}}, \frac{-g_{31}}{s_{\beta'}}\right), \text{ Or simply } \Theta 44 = 180 + \Theta 5$$

$$\theta_{66} = \gamma = atan2\left(\frac{g_{23}}{s_{\beta'}}, \frac{g_{13}}{s_{\beta'}}\right), \text{ Or simply } \Theta 66 = 180 + \Theta 6$$

Now, if β = 0 or 180, this means that the robot in a singular configuration where the joint axes 4 and 6 are parallel to each other. This results in a similar motion of the last three intersection links of the robot manipulator.

Alternatively:

- If $\beta = \Theta_5 = 0$, the solution will be
  $\Theta_4 = \alpha = 0$
  $\Theta 6 = \gamma = \text{atan2}(-g12, g11)$

- If $\beta = \Theta_5 = 180$, the solution will be
  $\Theta_4 = \alpha = 0$
  $\Theta 6 = \gamma = \text{atan2}(g12, -g11)$

MATLAB code for inverse kinematic analysis is attached to appendix

MACHINE VISION

INTRODUCTION:

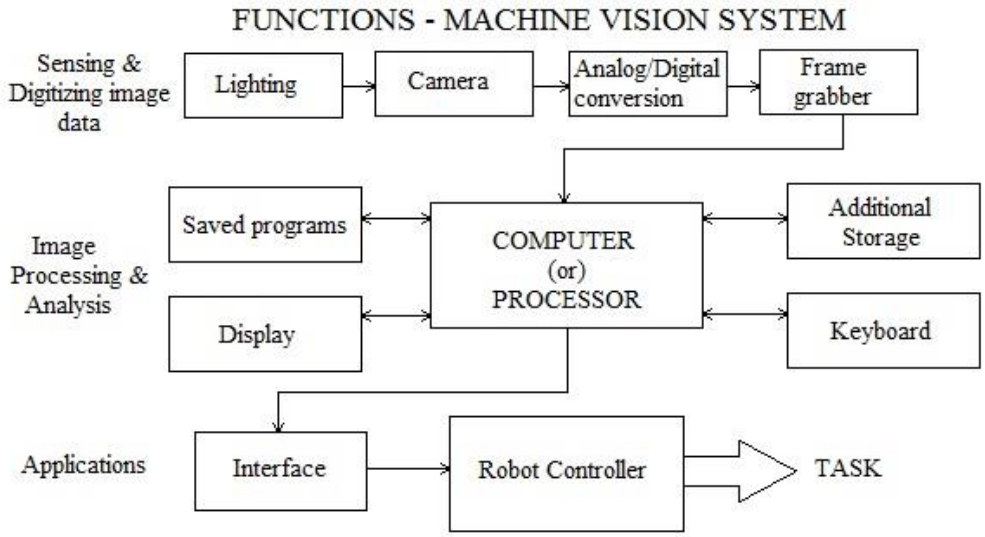
Machine vision enables a computer to recognize and evaluate images. It enables a computing device to inspect, evaluate and identify still or moving images.

Machine Vision Systems:

A machine vision system typically consists of digital cameras and back-end image processing hardware and software. The camera at the front end captures images from the environment or from a focused object and then sends them to the processing system. Depending on the design or need of the MVS, the captured images are either stored or processed accordingly.

For Robots Machine vision system is a sensor used for recognizing an object. It has several components such as a camera, digital computer, digitizing hardware, and an interface hardware & software. The machine vision process includes three important tasks:

- Sensing & Digitizing Image Data
- Image Processing & Analysis
- Applications



Tasks Performed by Machine Vision Systems

Sensing & Digitizing Image Data:

A camera is used in the sensing and digitizing tasks for viewing the images. It will make use of special lighting methods for gaining better picture. These

images are changed into the digital form known as the frame of the vision data. A frame grabber is incorporated for taking digitized image continuously.
Instead of scene projections, every frame is divided as a matrix. By performing sampling operation on the image, the number of pixels can be identified. The pixels are generally described by the elements of the matrix. A pixel is decreased to a value for measuring the intensity of light. As a result of this process, the intensity of every pixel is changed into the digital value and stored in the computer's memory.

Image Processing & Analysis:

In this function, the image interpretation and data reduction processes are done. The threshold of an image frame is developed as a binary image for reducing the data. The data reduction will help in converting the frame from raw image data to the feature value data. The feature value data can be calculated via computer programming. This is performed by matching the image descriptors like size and appearance with the previously stored data on the computer.

The image processing and analysis function will be made more effective by training the machine vision system regularly. There are several data collected in the training process. Here, the camera will be very helpful to identify the match between the computer models and new objects of feature value data.
Important applications of the machine vision system in the robots are:

- Inspection
- Localization
- Detection

Object detection is one the most important applications of machine vision. It is the process of finding instances of real-world objects in images or videos. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category. Some interesting applications of object detection are:

- Security
- Tracking objects
- Video surveillance

- Self-driving cars
- Face detection

The main aim of Object Detection is to identify the objects on the basis of following factors:

- Color
- Size
- Texture

So our aim is to make the camera identify the objects accurately. So that Robotic Arm is instructed accordingly to pick the objects from conveyer belt and place it onto instructed location.

HARDWARE:

The robotic arm is the primary hardware we are using. Camera is responsible for feature detection and controlling the arm. The plan is to attach the camera on top of the arm and then control the arm to pick up the object detected by the camera. The hardware used for object detection are as follows-

1. UArm swift pro robotic arm
2. OpenMV M7 Camera
3. Jumper wires (for interfacing)
4. 3-D Printer Interface Board
5. SD-Card

OpenMV M7 Camera:

The OpenMV Camera is a small, low power, microcontroller board which allows you to easily implement applications using machine vision in the real-world. You can program the OpenMV Cam in high level Python scripts.

The module has STM32F765VI ARM Cortex M7 processor running at 216 MHz with 512KB of RAM and 2 MB of flash. All I/O pins output 3.3V and are 5V tolerant.

The OV7725 image sensor is capable of taking 640x480 8-bit Grayscale images or 640x480 16-bit RGB565 images at 60 FPS when the resolution is above 320x240 and 120 FPS when it is below. Most simple algorithms will run at above 30 FPS. The module has a separate IDE for programming the board through Computer. It uses micro python for that purpose.

Hardware Module OpenMV M7

Some Applications where this module can be used are as follows:

- Face Detection
- Color Tracking
- Marker Tracking
- QR Code Detection
- Image Capture
- Video Recording

The Module has a separate slot for the SD-card.

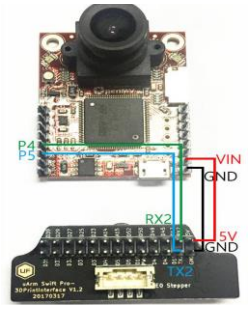
SD-card used by is 16-GB from SanDisk. The card is used to store the code and models so that the module can be used independently with the arm.

3-D Printer Interface Board came along with the arm. This board provides pins in order to connect the camera with arm using Jumper Wires.

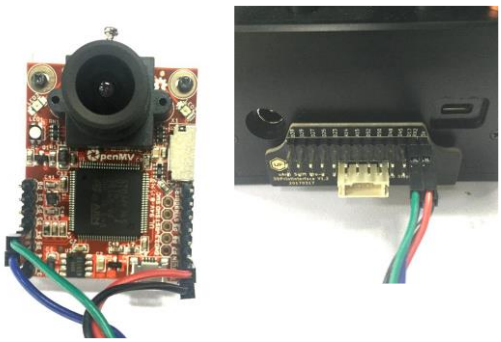
INSTALL CAMERA ON THE ARM

Using all the hardware available we made the connections between the UArm and OpenMV M7 Camera.

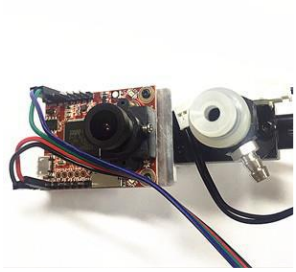
1) Connecting the Respective Pins with Jumper Wires.



2) Connecting the interface Board with the ARM



3) Mounting the camera Module on the end-effector



The installation is done on the ARM. Now all we have to do is connect the ARM to the computer and open the serial monitor of the Arduino IDE adjust the settings (newline & 115200 baud) and then send the G-code Command M2500 which will switch the main UART port from USB to the port of OpenMV.

METHODS FOR OBJECT DETECTION

There are a lot of techniques available for object detection. The technique that we will be using is chosen considering the following factors:

- The object detection algorithm used should be fast as well as accurate because the object as well as the ARM will be moving.
- The algorithm should be easy to train and the training time should be also less.
- The algorithm used should be dynamic so that we can make changes accordingly.
- The Object Detection Model used should be reliable.

The Different Object Detection methods tested are as follows:

- Training the Pre-Trained Models like faster_RCNN, Inception and MobileNet with SSD using the object detection API.
- Feature-based object detection using SURF and FLANN+KNN.
- Contour+Descriptor Extraction Based algorithm known as merger which uses RGB-D sensors.
- Object Tracking and Recognition Algorithms coded in OpenMV IDE which uses only the built in libraries of OpenMV IDE.
- Training a custom neural network using Caffe on a PC and deploying the network on the OpenMV Cam.

The results and observations for the above methods are discussed here.

1. USING OBJECT DETECTION API

TensorFlow Object Detection API is an open source framework built on top of TensorFlow that make it easy to construct, train and deploy object detection models. We have used this API to train some pre-trained models on our dataset and compare which of the models fits best with our project that is meet the factors discussed above. The models which we trained are ssd_mobilenet_v3, ssd_inception_v2 and faster_rcnn_pnas on our image dataset. We referred to the following link for training and deploying the model:
https://3sidedcube.com/guide-retraining-object-detection-models-tensorflow/
https://github.com/tensorflow/models/tree/master/research/object_detection/models

The computation time for ssd with Mobile net was the least out the three, Inception took the most time out of all and as for the accuracy all the models

were able to detect correctly based on our dataset. So the deciding factor for us was time only because else all of them were tensorflow based models and were trained on the same dataset .So the model which is the fastest of them is preferred. So we decided to move on with tensor flow model of mobilenet_ssd_v3. SSD with MobileNet refers to a model where model architecture is SSD (Single Shot Detector) and the feature extractor type is MobileNet.

All these computation were performed on the PC, it is not yet tested it with the OpenMV M7 camera. So on testing the model SSD with Mobile net on OpenMV M7 camera the results were not desirable because the model took a lot of time to run on the camera as the computation power of the camera is very less and also the libraries used inside the model were not available there on the OpenMV IDE so I had to add it manually as it is open source software. The process for testing the model was very complex and very time consuming.

2. SURF and FLANN

Here we have used SURF algorithm and the FLANN based matcher (Fast Library for Approximate Nearest Neighbors), and their OpenCV 3.0 CPU implementations. The SURF algorithm falls into Feature-based object detection.

Feature Extraction with SURF:

SURF stands for Speeded up Robust Features and is an algorithm which extracts some unique key-points and descriptors from an image. A set of SURF key-points and descriptors can be extracted from an image and then used later to detect the same image. SURF uses an intermediate image representation called Integral Image, which is computed from the input image and is used to speed up the calculations in any rectangular area. It is formed by summing up the pixel values of the x, y co-ordinates from origin to the end of the image. This makes computation time invariant to change in size and is particularly useful while encountering large images. The SURF detector is based on the determinant of the Hessian matrix. Basically the features of SURF algorithm are:

- Find key-points in the image using Hessian matrices
- Determine the orientation of key-points and draw a square region around it.

- Use Haar wavelets in oriented square region around the key-points to find intensity gradients in the X, Y direction. As the square region is divided into 16 squares, and each such sub-square yields 4 features, therefore SURF descriptor for every interest point is 64 dimensional.

Feature Matching with FLANN (Fast Library for Approximate Nearest Neighbors):

After unique key-point and descriptors are extracted from both images object and scene, a matching must be done. We implement FLANN which is also a matching strategy. The FLANN library contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. We have used the k-nearest neighbors algorithm which uses DescriptorMatcher::knnMatch () which returns the best k matches.

Code for the detection using SURF and FLANN is included in the Appendix.
Code searches for on object inside scene using the SURF for key-points and descriptors detection, and FLANN+KNN for matching them.

3. CONTOUR + DESCRIPTOR EXTRACTION USING SURF AND FLANN

Contour extraction and descriptor extraction are explained below.

1. Contour extraction:

Contour are the shapes of the object which are used to determine what the given item is. For contour extraction, Kinect camera is used as it is the combination of infrared and RGBD camera which gives gray scale image indicating how far the pixels are from the point of observer. After contour is extracted OpenCV is used to find correlation amongst other. One of the disadvantage is that, given many objects, it is difficult to distinguish between two different objects having similar contour e.g. apple and orange. While training, image is converted into matrices consisting of some finite rows and columns where each value are used for correlation. Pixel from the image to be processed are matched with trained matrix giving correlation matrix which is then used to detect the object.

Initially, Image obtained from Kinect image is equalized and converted to Mat type and then morphological transformations are implemented to enhance the information needed. Gray value indicated the distance and using them closer objects are extracted. As robot moves, object can be limited to a certain distances

only. Now, code calculated which pixels form biggest contour and then correlation is applied to the model.

Implemented Contour Extraction Algorithm

```
openCV_formatimage = convertOpenCV (image received)
image = convert2OpenCV (BLACK WHITE CODE; image)
image = openCV_ Equalize(image)
MatFormat_image_2 = convertMatFormat(image)
Erode(image 2)
Dilate(image 2)
for pixel i do //This if for all pixels in the image
if(isTooFar(pixeli)) then
pixel_i = 0
else
pixel_i = 1
end if
end for
contour_selected ==  getBiggerContour(arrayContours)
getBoundigBox(contour selected)
correlate(image 2; models; results)
Point extrema = getExtrema(results)
if ((extrema:x > extrema:y)OR(extrema:x > extrema:z)OR(rangeFullFills())) then
print("ObjectDetected!")
end if
```

2. Descriptor Concept:

Image processing is difficult because of constantly changing scenarios and having to add new object for detection. But even if image is taken from same position it is difficult to match the same images because of illumination and hence finding an object became and increasingly tough job. Hence, we need to extract certain features from the image which can be used to match image irrespective of other factors and this process of extracting important feature is called feature extraction. To perform feature extraction, first key points are extracted from different unique locations and then neighbor region of each key-points are represented by feature vector and unique feature descriptor are computed for each region. This feature descriptor are unique and robust to noises. The matching of these descriptors is based on a distance between the vectors to find if object is detected.

Implemented Descriptor Algorithm

```
keyPoints = KPextraction(image received)
matches[] = FLANN Match Model(image received)
sum = 0
for i = 0; i < matches:size(); i + +) do
if (distance(matches[i]) < DISTANCE PREFIX) then
sum+ = distance(matches[i])
end if
end for
if (sum > EMPIRICAL DATA) then
print("ObjectDetected!")
end if
```

Proposed Algorithm

For our project we used the merger technique. It implies that results from both the algorithm will be used to determine the object.

Given all the different algorithms, we decided to implement our algorithm using SURF (Speed up robust feature) and FLANN (Fast approximate nearest neighbor search) which revolves mainly around the concept of contour extraction and descriptor extraction concept respectively and as we have implemented it already.

Important Points regarding this method:

- It will generalize the contour and without proper and unique descriptor it might give incorrect output.
- Descriptor part of detection is hard because SURF and FLANN might be able to find good matches even if the object is not present.
- Also, adding an object in the model is simple as it takes minor changes in code rather than retraining whole model completely but with adding of objects, margin of errors only increases.

To implement the algorithm we have to use the OpenCV library or I have to add the functions SURF and FLANN manually as OpenMV Software is open Source.

4. USING BUILT IN LIBRARIES OF OPEN MV IDE

It features a powerful text editor powered by QtCreator, a frame buffer viewer, histogram display, and an integrated serial terminal for debug output from the OpenMV Cam.

- It has a generic micro python board support.
- It has preinstalled codes for line detection, video recording, and face detection.
- The camera supports a SD card which can be used to store the code.

The basic classification techniques are already available in the IDE .So we combined the different function in IDE and implemented object tracking, Image Matching, Find_Contour, Find_Keypoints, Color Tracking. We have used functions like:

- Find_Blobs
- Template_Matching
- Sensor.Snapshot
- Img.draw_rectangle

This is the sample code for finding the first rectangle object from a scene.

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA)
sensor.skip_frames(time = 2000)
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()

    for r in img.find_rects(threshold = 10000):
        img.draw_rectangle(r.rect(), color = (255, 0, 0))
        for p in r.corners(): img.draw_circle(p[0], p[1], 5, color = (0, 255, 0))
        print(r)

    print("FPS %f" % clock.fps())
```

This method is fast and not at all complex and also adding objects is also easy as we have to only add an extra template_matching command in the code along with the path of the image captured through the camera but the functions alone are not enough to accurately identify the images because there are not enough function available that we can operate on images so that we can get desired results for example we need erode and dilate function in order to sharpen a image obtained from mat format but these are not available in the IDE. Likewise there are a lot of functions that are still not there in IDE which are necessary for accurate object detection.

5. CMSIS-NN Library:

The CMSIS-NN library brings deep learning to low-power microcontrollers such as the Cortex-M7 based OpenMV camera.
The default CMSIS-NN library comes with a CNN example trained on the CIFAR-10 dataset. However, this example is hard-coded, meaning it must compiled and linked with the main application. It allows users to convert Caffe models to a quantized binary format which can be loaded from the file-system (SD Card or internal flash) at run-time. Additionally, it takes care of preprocessing the input image, subtracting the mean, and scaling the data if required.

We trained the CIFAR-10 Model on Object dataset available at ImageNet. The model should be strictly in Caffe Library. The first step after training the network is to use the quantization script provided by ARM to convert the Caffe model weights and activations from floating point to fixed point format.
The output of this script is a serialized Python (.pkl) file which includes the network's model, quantized weights and activations, and the quantization format of each layer. Running this command generates the quantized model:

python2 nn_quantizer.py –model models/obj/obj_train_test.prototxt –weightsmodels/obj/obj_iter_*.caffemodel—savemodels/obj/obj.pkl

The next step is to use our NN converter script to convert the model into a binary format runnable by the OpenMV Cam. The converter script outputs a code for each layer type followed by the layer's dimensions and weights.
Running this command generates the binary model:
python2 nn_convert.py—model models/obj/obj.pkl--mean/path/to/mean.binaryproto --output obj.network

After this we deploy the model onto the camera that is the SD card and run it on the computer through the IDE. For Deploying we use the already available codes of feature detection cnn.py and keypoint_save.py so that we can extract descriptors from the images and pass the region of interest (ROI) to the CNN to detect Objects. The first part of the Object detection loads the network into memory .It's possible to slide the detection window over the entire image doing so would be very slow.

```
# Load Object Detection network
net = nn.load('/obj.network')

# Load CNN.py
feature_detect = image.feature_detect ("frontalface", stages=25)
print(feature_detect)
#The next step is capturing a snapshot and finding all the objects.

# Capture snapshot
img = sensor.snapshot()
# Find descriptors
objects=img.find_features(feature_detect, threshold=0.75, scale_factor=1.25)
#Finally, for each detected object, the region of interest is slightly cropped and
passed to the neural network. Note that the object detection network is trained
on tightly cropped faces so we have to reduce the size of the ROI.

# Detect objects
for r in objects:
  # Resize and center detection area
  r = [r[0]+10, r[1]+25, int(r[2]*0.70), int(r[2]*0.70)]
  out = net.forward(img, roi=r, softmax=True)
  img.draw_string(r[0], r[1], ':)' if (out[0] > 0.8) else ':(', color=0, scale=2)
```

Lenet model also worked and actually it was very fast as compared to the rest of the models which we have tested so far and the training time was also less as the no. of images were less compared to YOLO and MobileNet.
This is the best Method we have found so far for the machine vision which we have implemented using OpenMV M7 camera.

CONCLUSION

As we have discussed all the methods that were tested by us. The conclusions come out as for machine vision we need algorithms that works fast and have low complexity so that the computation needed is less because as the computation increases the time increases to perform operations. So out of all the techniques discussed the CMSIS-NN Library method was the most suitable.

As in CMSIS-NN library too also works only with the models which are trained in caffe but this method is reliable because this gives accurate as well as fast results and soon the founders of OpenMV IDE are going to release networks corresponding to models like YOLO, Mobilenets. So this method is meeting all

the factors that were discussed in the starting of Machine Vision. The CIFAR-10 model is already loaded on the Camera .It has to interface to the Arduino-Mega 2560 Board inside the ARM through UART. So that the working port of the ARM can be transferred from USB to UART through the computer. We can also train models like Lenet which is used to recognize numbers.

GESTURE CONTROL:

INTRODUCTION:

Gesture is a movement of part of a body (mostly hand/head) to convey a meaning or express an idea. In a robotic arm, gesture control refers to controlling of the robotic arm using gestures made by human hands, legs, head etc. So, the entire project is about controlling the arm movement using hand gesture movements. Hence the robotic arm simply replicates our hand movements and moves accordingly. However, the project is to control the gestures of the robotic arm wirelessly i.e. there is no direct connection or link between the hand and the robot.

HARDWARE:

There are various hardware components used in the project. The hardware includes-

1. One Arduino Nano
2. One Arduino Uno
3. Two Nodemcu modules
4. One MPU6050 module
5. Robotic arm
6. Jumper wires (all kinds)
7. Soldering kit
8. Bread boards
9. Wi-Fi router

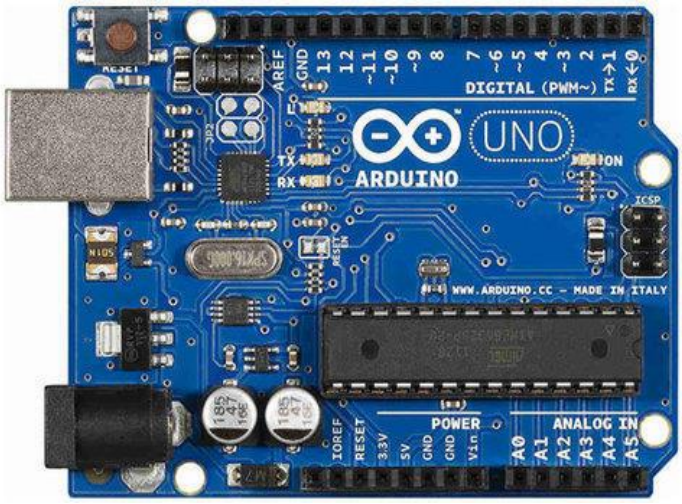
So for the ease of experimentation, the whole hardware is divided into two sections because the communication is wireless. One is called as sender circuit and other one is called is called the receiver circuit. In the sender circuit, we have one Arduino Nano, MPU6050 module, jumper wires and one Nodemcu. In the receiver circuit, we have one Arduino Uno, Robotic arm, one Nodemcu, jumper wires and bread board (optional).

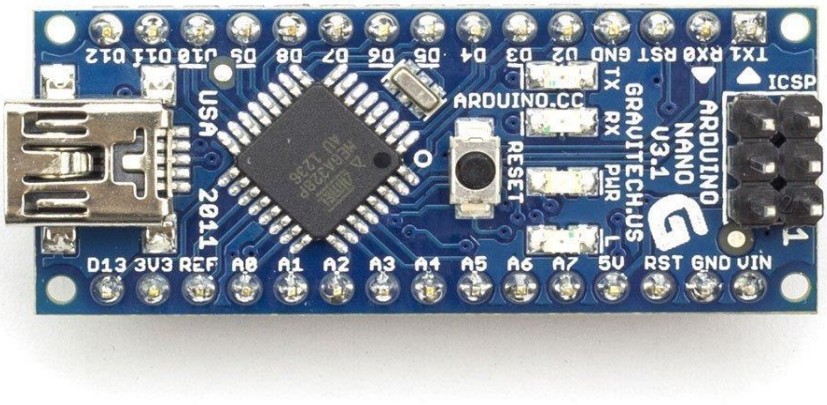
Now, let us look into each one of them, their general uses and features. Also, why these particular modules are chosen and the purpose they serve in the project.

(1) ARDUINO:

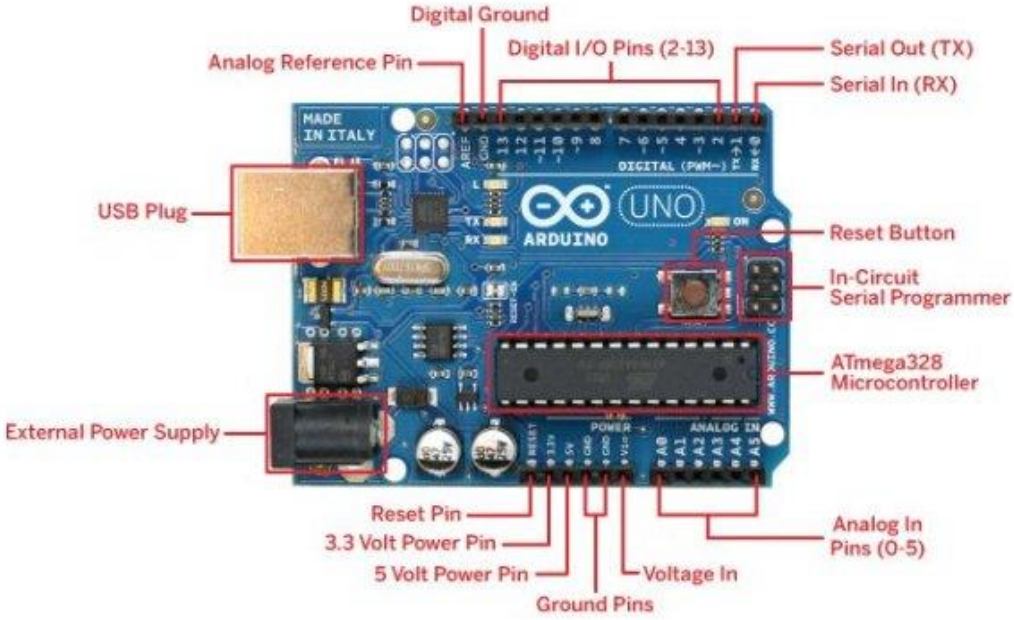
The Arduino is an open source hardware and software company which manufactures a single-board microcontroller and microcontroller kits for building digital devices or interactive objects. It is a physical programmable circuit and can be used to program any digital or analog devices. The microcontroller used in it is atmega series. It has a voltage output of 5V and 3.3V, analog and data pins to send and receive signals.

Arduino has many different models out of which only Uno and Nano are used. Arduino Uno has more pins for a better interface and Arduino Nano has less size in comparison to Uno. So, Nano is used in the sender circuit and the Uno is used in the receiver circuit due to its size advantage. The Arduino here is the programmable device and can be used to program the robot, MPU6050 and the Nodemcu modules.

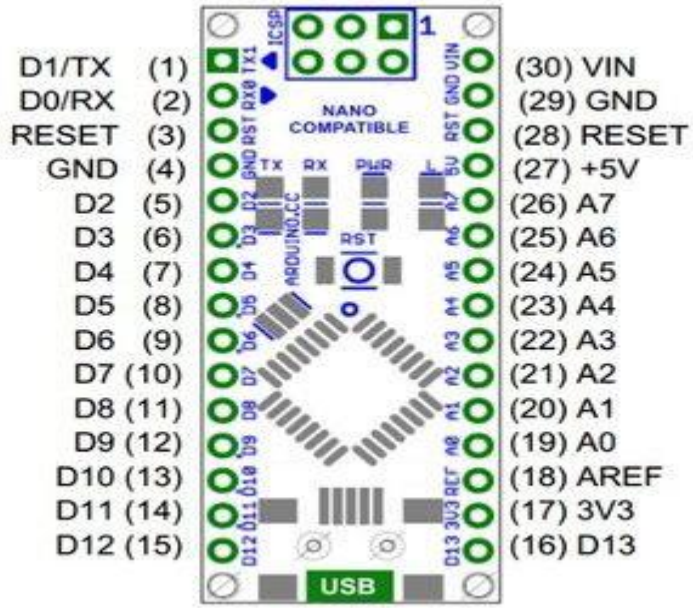




Pin description of Arduino:



Arduino Uno has 6 analog pins, 14 data pins (digital), three ground pins, Two 5V pins, one 3.3V pin, a pair of Tx and Rx pins.

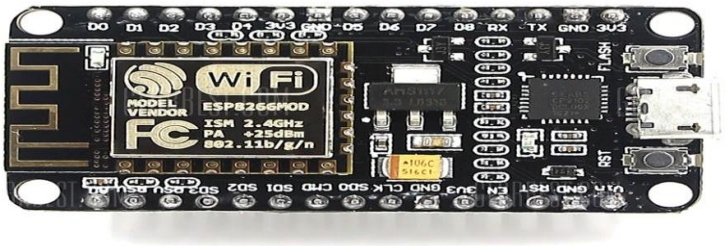


Arduino Nano has 8 analog pins, 14 digital pins, two ground pins, two 5V pins, one 3.3V pin, a pair of Tx and Rx pins.

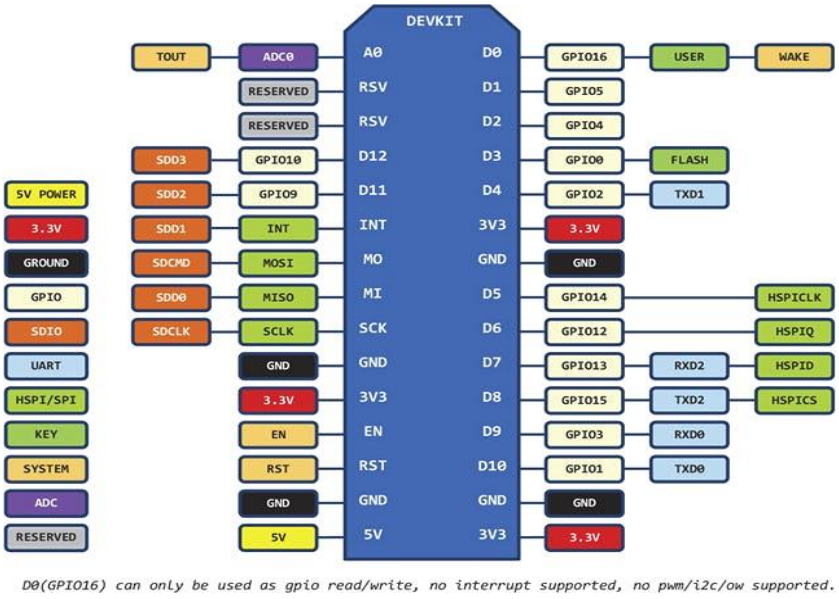
(2) NODEMCU and Wi-Fi ROUTER:

Nodemcu is a very common device in the series of ESP8266 Wi-Fi module. It is also a programmable device and has an inbuilt microcontroller. It has a working voltage of 3.3V and has an inbuilt level converter from 5V to 3.3V. The Nodemcu board is in general used to host internet servers and can also be used to connect to Wi-Fi and establish connections using both I2C and SPI protocols.

In the project, Nodemcu is used for data transmission to server and data receiving from server using a Wi-Fi router. One module out of two transmits the sensor data to the server and other module receives it from the server. Hence, the data is conveniently sent from one module to other module using a Wi-Fi router. Hence these are used for the purpose of communication



Pin description:



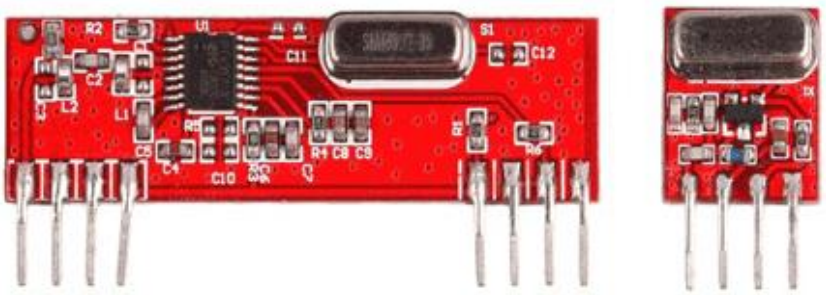
Pin Names on NodeMCU Development Kit	ESP8266 Internal GPIO Pin number
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD2	GPIO9
D12/SD3	GPIO10

(3) OTHER ALTERNATIVE:

The previous part provides the information about Nodemcu that was used in achieving communication between two Arduinos. But, there are some other hardware that can replace these Nodemcu. They are Radio frequency (RF) transceiver circuit and the data encoder decoder circuits.

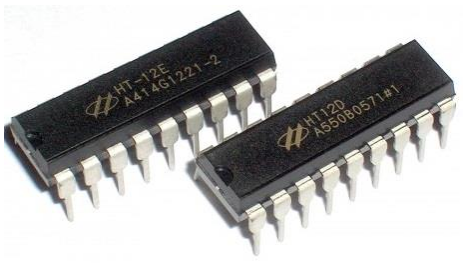
(1) RF Transmitter and Receiver circuits:

The RF transceiver circuit can be used in place of Nodemcu modules for communication. RF circuits use radio frequency waves and transmit the data from the transmitter end to the receiver end. The only problem using them is that data sent and received must be in the encoded form. Hence, we need to encode the data before sending and decode it later using same encoding algorithms. In the below figure, left one is the RF transmitter and the right one is the RF receiver. RF Transmitter must be used in sender circuit and RF Receiver in the receiver circuit.



(2) Encoder and decoder circuits:

The encoder and decoder circuits are used to encode the data before transmission and decode the data after receiving. The most popularly used encoder and decoder circuits are HT-12E (encoder) and HT-12D (decoder) as shown below. HT-12E converts 4-bit parallel data into serial data in order to transmit it over the RF link. Similarly HT-12D converts serial data received into 4-bit parallel data. HT-12E must be on sender circuits and HT-12D must be on receiver circuits.

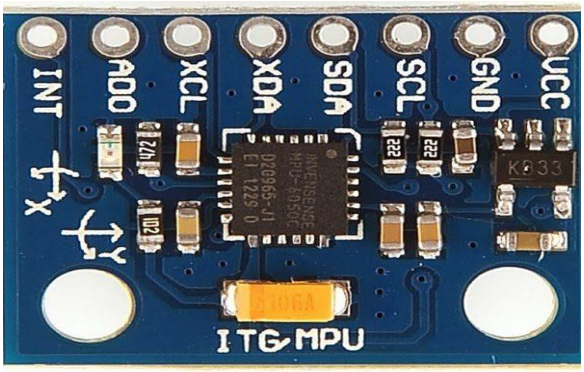


(4) MPU6050:

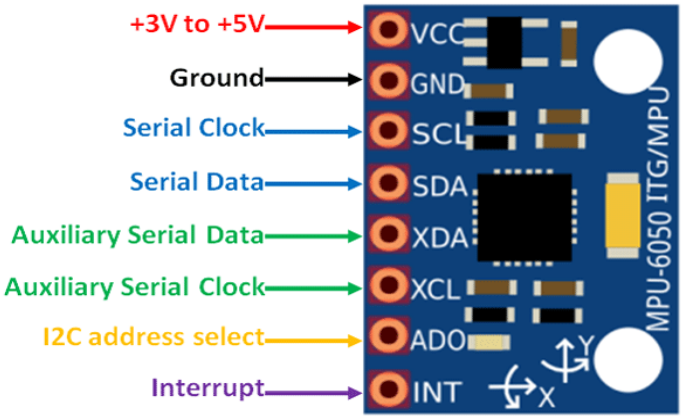
MPU6050 is a module which has a three axis accelerometer, three axis gyroscope and an inbuilt temperature sensor. It can be connected externally to a three axis magnetometer to make it a nine axis motion tracking device. Accelerometer senses the acceleration values using gravity calculations, gyroscope senses the angular momentum of the device. These sensors data can be used to program the Arduino for the motors to work accordingly. The working voltage value is 5V. So, it can be directly connected to Arduino without the use of voltage level shifters.

In the project, MPU6050 module is used in the sender circuit because it is the sensor which gives the output data to the Arduino. This module is selected because it has large range and it is efficient. The values can be mapped easily and can be used in the program for motor control.

In the project, we only need the accelerometer values. We don't need the values of gyroscope and the temperature. So, we program the Arduino in such a way that only accelerometer data is sent over the server via Nodemcu.



Pin Description:



(5) ROBOTIC ARM :

One arm prototype was given to us for the experimentation and reference purposes. The robotic arm was U-Arm Swift Pro. These two arms were given and we were asked to look into its design and other aspects as a reference for building an arm. Both have their own advantages and benefits.

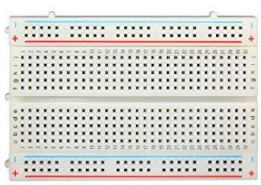
U-Arm Swift Pro

(6) OTHER INTERFACING HARDWARE :

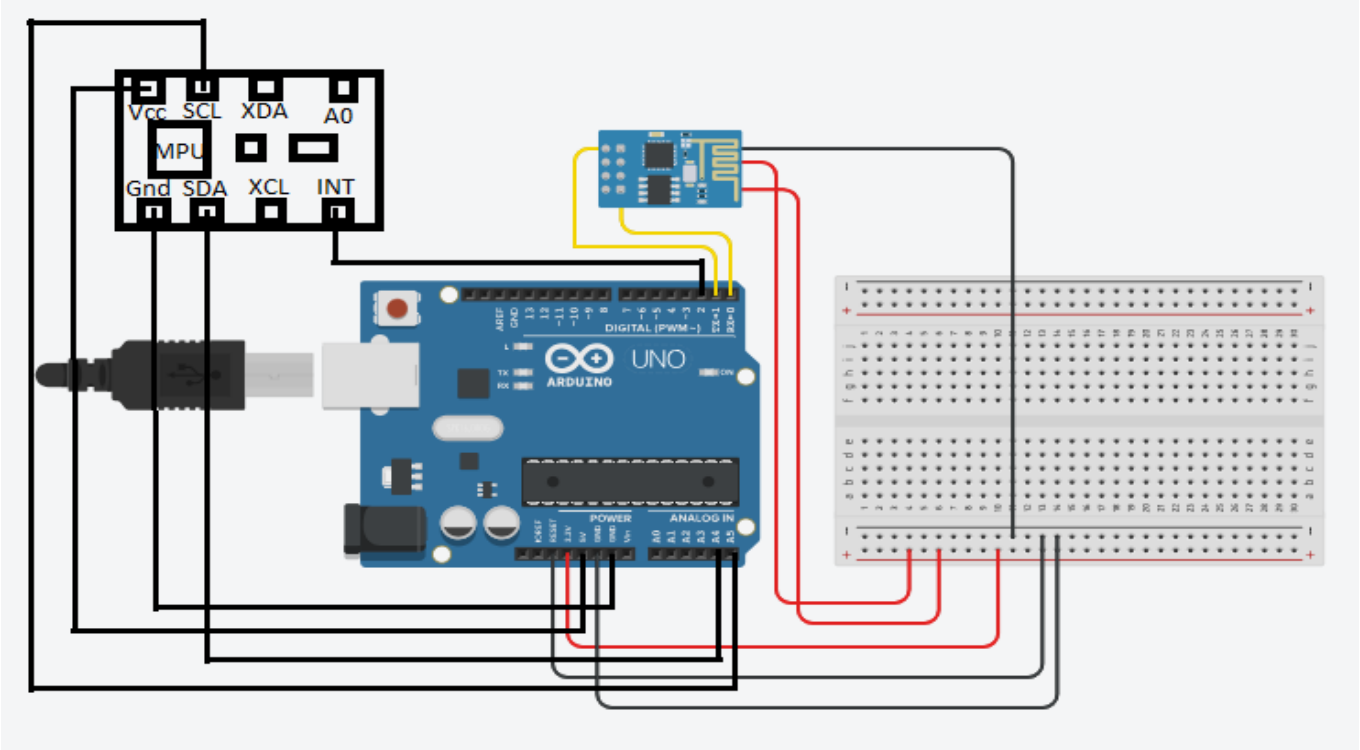
Other interfacing hardware includes breadboard, jumper wires and soldering kit. The breadboards come in various sizes and we use a mini breadboard for the project due to its less size.

We have three different kinds of jumper wires namely male-male, male-female and female-female connecting wires. These can be used to connect different hardware parts temporarily.

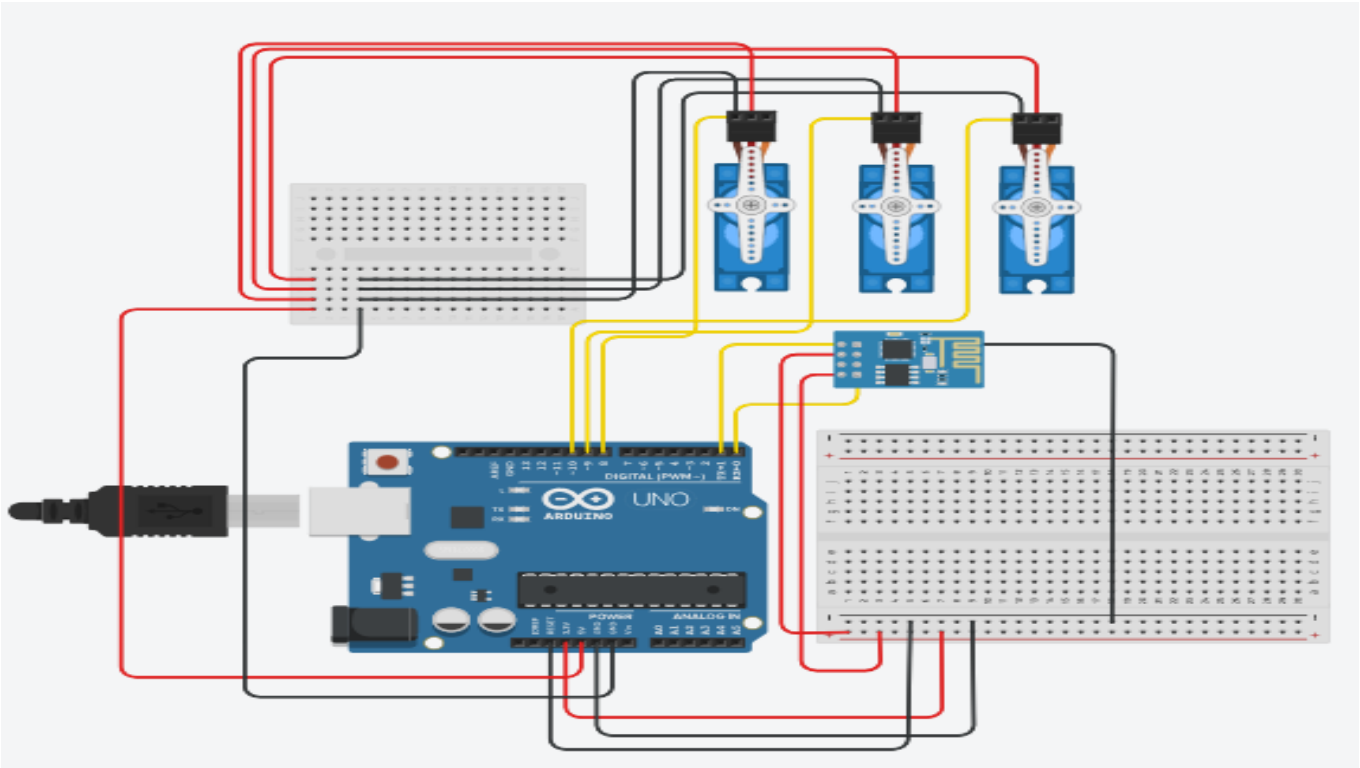
Soldering kit is used in case if there is any connection is needed to be made a permanent one. It is used to attach pins to many devices.



HARDWARE CONNECTIONS:



SENDER CIRCUIT



RECEIVER CIRCUIT

CONNECTION DESCRIPTIONS:

SENDER CIRCUIT:

MPU6050	-	Arduino
Vcc	-	5V
INT	-	D2
SDA	-	A4
SCL	-	A5
GND	-	GND

Nodemcu	-	Arduino
Vin	-	Vin
GND	-	GND
Tx	-	D3
Rx	-	D2

RECEIVER CIRCUIT:

Servo	-	Arduino
SIGNAL	-	D9, D10, D11
Vcc	-	5V
GND	-	GND

Nodemcu	-	Arduino
Vin	-	Vin
GND	-	GND
Tx	-	D3
Rx	-	D2

HARDWARE COST OUTLOOK:

Cost of hardware also plays an important role in robotics. The price of hardware used also determines the final price of fully developed model. So, let us look into the cost of each hardware used in the project.

Hardware	**Quantity**	**Price per unit**	**Total price**
Arduino Uno	1	500/-	500/-
Arduino Nano	1	350/-	350/-
MPU6050	1	250/-	250/-
Nodemcu	2	365/-	730/-
Soldering kit	1	500/-	500/-
Jumper wires (all)	3	160/-	480/-
Bread board	2	160/-	320/-

*Latest prices courtesy in INR– Amazon.in

Overall cost excluding robot making and Wi-Fi router = 3130/-

These are the cost details when we use RF transceiver and encoder-decoder circuits instead of nodemcu and Wi-Fi router.

Hardware	Quantity	Price per unit	Total price
Arduino Uno	1	500/-	500/-
Arduino Nano	1	350/-	350/-
MPU6050	1	250/-	250/-
RF transceiver	1	500/-	500/-
Encoder-Decoder	1	190/-	190/-
Soldering kit	1	500/-	500/-
Jumper wires (all)	3	160/-	480/-
Bread board	2	160/-	320/-

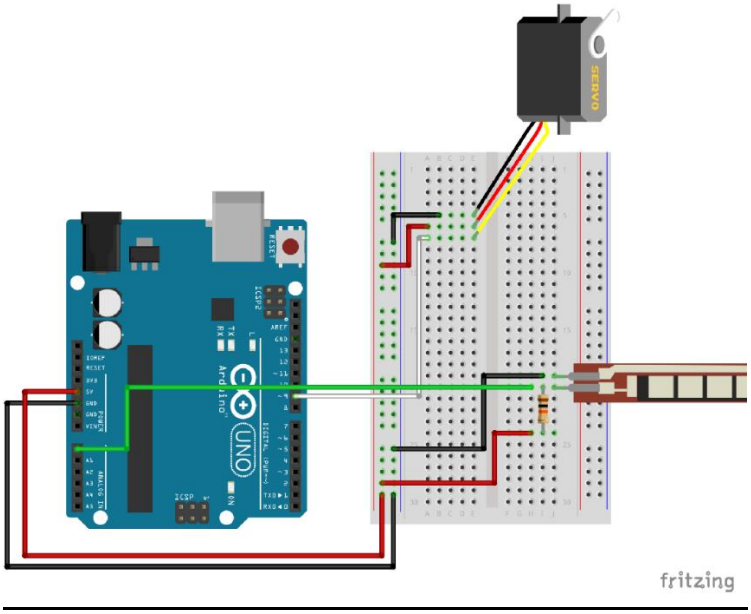
*Latest price courtesy in INR – Amazon.in

Overall cost excluding robot making = 3090/-

ADDITIONAL HARDWARE FOR ARM GRIPPER:

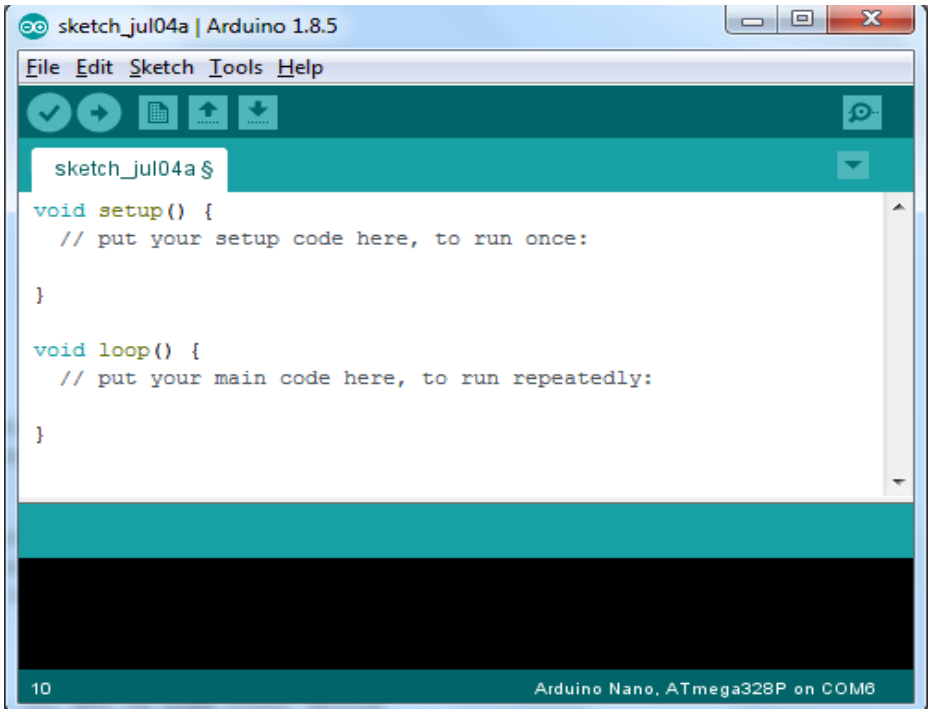
The robotic arm can also be equipped with a gripper which can hold objects. This part is excluded in the circuit and in the implementation but is worth giving a try. For the gripper movement, the most suitable hardware is the flex sensors.

These flex sensors work on principle of increasing and decreasing resistances with its nature of bending. When the flex sensors are bent they have a more resistance and lesser resistance when they are straight. By this principle, one can arrange it in such a way between index and thumb such than the servo moves as soon as the flex sensor is bent. The increase in resistance caused by the bending can be converted to voltage difference increase which can make the servo move. Below mentioned figure is the connection of flex sensor, Arduino and the servo motor. Hence this flex sensor must be place on the sensor circuit because it is also transmitting the data to Nodemcu for servo movement. So, connect the circuit as shown below and then upload the code into the Arduino. The Arduino code for servo movement using flex sensor is given in the appendix.



SOFTWARE:

Since we have seen the hardware requirements of the project, let us see the software face of the project. There are two programmable devices in the project which are Arduino and Nodemcu. However, we are not using Nodemcu anywhere for programming purpose but it is utilized in the communication using Wi-Fi. So, the only left programmable device is Arduino. We use Arduino IDE for Arduino programming, which is the best IDE available.



We are using two different Arduino in the project but the code is still the same regardless of the Arduino type. So, similar to the hardware, we have two different Arduino codes for two Arduinos. One code is the sender code and other code is the receiver code.

(1) Sender code:

The sender code is used for receiving data from MPU6050 and transmit it over the Web server. Most of the MPU6050 programming and the Nodemcu server programming are written in the sender code.

(2) Receiver code:

The receiver code is used to receive the data from the Web server and transmit it to corresponding motors for motion. Most of the servo related programming and Nodemcu client programming are written in the receiver code.

WORKING ALGORTHMS:

Sender Algorithm:

Step 1: Start.

Step 2: Start both ESP devices for communication.

Step 3: Initiate the communication via local Wi-Fi and establish Connection with receiver ESP device.

Step 4:

4.1: If connected, proceed to Step 5.

4.2: Else, proceed to Step 3.

Step 5: Start to take readings from MPU6050 (accelerometer).

Step 6: Send the data over the server using sender ESP device.

Step 7: Proceed to Step 4.

Receiver Algorithm:

Step 1: Start.

Step 2: Start the ESP device for communication.

Step 3: Establish connection with sender ESP device.

Step 4:

4.1: If connected, proceed to Step 5.

4.2: Else, proceed to Step 3.

Step 5: Read and interpret the data sent by MPU6050 from the server.

Step 6: Use the data in program to send instructions to motors.

Step 7: Proceed to Step 3.

GESTURES FOR ARM MOTION:

(1) Stop gesture:

If the hand is still and is parallel to ground as shown in the below figure, the arm is said to be in no motion. So, it is in a stable state.

(2) Forward gesture:

If the hand is pushed down from the x-y plane along the negative z axis, the arm moves forward. The gesture shown below causes the forward movement in the arm.

(3) Backward gesture:

If the hand is pushed up from the x-y plane along the negative z axis, the arm moves backwards. The gesture shown below causes the backward movement in the arm.

(4) Lift gesture:

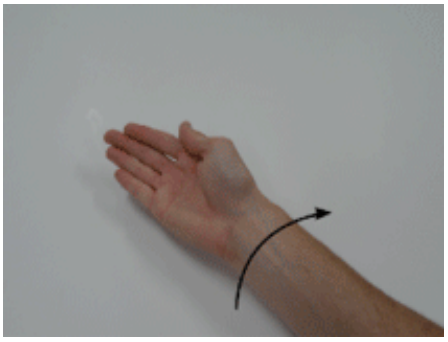
If the hand is placed in x-z axis and then moved along the positive z axis, the arm is lifted. The below shown gesture is used to lift the arm.

(5) Drop gesture:

If the hand is placed in x-z axis and then moved along the negative z axis, the arm is dropped. The below shown gesture is used to drop the arm.

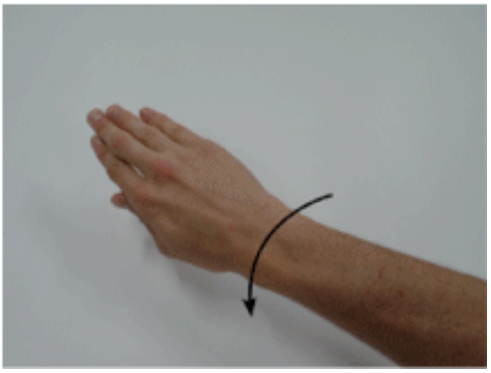
(6) Right move gesture:

If the hand is in the x-y plane and is tilted to the right along the x axis, the arm moves in right direction. The below shown gesture is used to move the arm in right direction.



(7) Left move gesture:

If the hand is in the x-y plane and is tilted to the left along the x axis, the arm moves in left direction. The below shown gesture is used to move the arm in the left direction.



ADVANTAGES:

There are many advantages of implementing and using gesture control. Some of them include:-

(1) Interactivity in real time with virtual objects.
(2) Machines don't get tired. They have no feelings and emotions and can even work in harsh and unstable situations unlike humans.
(3) Can be applicable in remote rural areas to carry out operations.

DISADVANTAGES:

Although there are many advantages and applications, there are some disadvantages also. Some of the disadvantages are:-

(1) Robots are not very suitable for making complex decisions.
(2) De-bugging issues in this would be complicated since they involve real life.
(3) Person might become lazy.

REAL LIFE APPLICATIONS OF GESTURE CONTROL:

There are many live present day applications for gesture controlled arm. Some of them include:-

(1) Remote control with hands is possible.
(2) Industrial applications for trolley control, lift control etc.
(3) Military applications to control robotics.
(4) Medical applications for surgery purpose.
(5) They are effective to help physically handicapped people.
(6) Construction and Civil applications.
(7) They can be used in gaming zones.
(8) They can be used to control toy cars and planes.
(9) They can be used in bomb disposal, so that human life is not at risk.

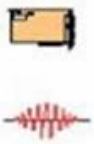
Voice Recognition

Introduction

Language is man's most important means of communication and speech its primary medium. Spoken interaction both between human interlocutors and between humans and machines is inescapably embedded in the laws and conditions of Communication, which comprise the encoding and decoding of meaning as well as the mere transmission of messages over an acoustical channel. Here we deal with this interaction between the man and machine through synthesis and recognition applications.

Speech Recognition is the ability of a computer to recognize general, naturally flowing utterances from a wide variety of users. It recognizes the caller's answers to move along the flow of the call.

Emphasis is given on the modeling of speech units and grammar on the basis of Hidden Markov Model & Neural Networks. Speech Recognition allows you to provide input to an application with your voice. The applications and limitations on this subject enlighten the impact of speech processing in our modern technical field. While there is still much room for improvement, current speech recognition systems have remarkable performance. We are only humans, but as we develop this technology and build remarkable changes we attain certain achievements. Rather than asking what is still deficient, we ask instead what should be done to make it efficient.

USER	MICROPHONE	SOUND CARD	SPEECH RECOGNITION ENGINE	SPEECH AWARE APPLICATION
			W AA TD T AY M IH S IH IT	WHAT TIME IS IT
User speaks into the microphone.	Captures sound waves & then generate electrical impulse	Converts acoustic signal to digital signal	Converts digital signal to phonemes, then words	Application process words as text input.

Speech Recognition Algorithms:

Dynamic Time Warping:

One of the earliest approaches to isolated word speech recognition was to store a prototypical version of each word (called a template) in the vocabulary and compare incoming speech with each word, taking the closest match. This presents two problems: what form do the templates take and how are they compared to incoming signals.

The simplest form for a template is a sequence of feature vectors -- that is the same form as the incoming speech. We will assume this kind of template for the remainder of this discussion. The template is a single utterance of the word selected to be typical by some process; for example, by choosing the template which best matches a cohort of training utterances.

Comparing the template with incoming speech might be achieved via a pairwise comparison of the feature vectors in each. The total distance between the sequences would be the sum or the mean of the individual distances between feature vectors. The problem with this approach is that if a constant window spacing is used, the lengths of the input and stored sequences is unlikely to be the same. Moreover, within a word, there will be variation in the length of individual phonemes: *Cassidy* might be uttered with a long /A/ and short final /i/ or with a short /A/ and long /i/. The matching process needs to compensate for length differences and take account of the non-linear nature of the length differences within the words.

The Dynamic Time Warping algorithm achieves this goal; it finds an optimal match between two sequences of feature vectors which allows for streached and compressed sections of the sequence.

Hidden Markov Model:

The Hidden Markov Model (HMM) is a relatively simple way to model sequential data. A *hidden* Markov model implies that the Markov Model underlying the data is hidden or unknown to you. More specifically, you only know observational data and not information about the states. In other words, there's a specific type of model that produces the data (a Markov Model) but you don't know what processes are producing it. You basically use your

knowledge of Markov Models to make an educated guess about the model's structure.

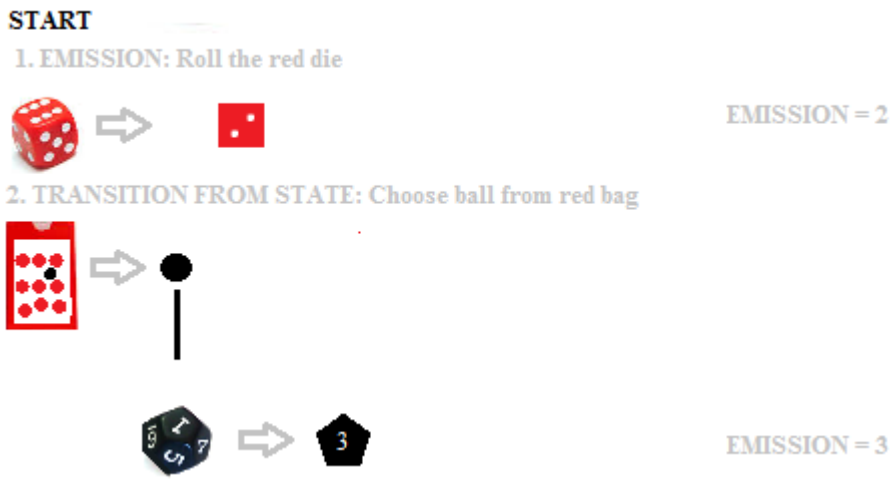
What is a Markov Model?

In order to uncover the Hidden Markov Model, you first have to understand what a Markov Model is in the first place. Here I'll create a simple example using two items that are very familiar in probability: dice and bags of colored balls.

The model components, which you'll use to create the random model, are:

- A six-sided red die.
- A ten-sided black die.
- A red bag with ten balls. Nine balls are red, one is black.
- A black bag with twenty balls. One ball is red, nineteen are black.

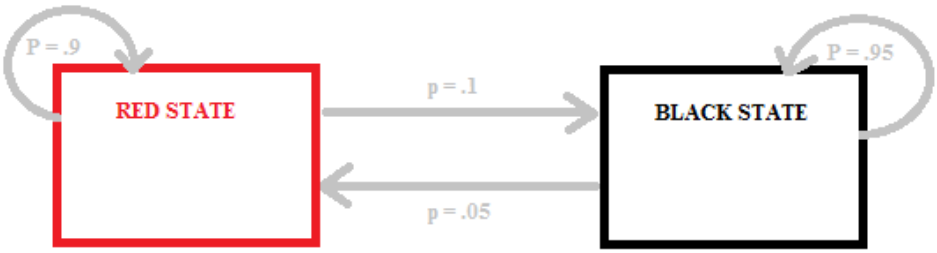
"Black" and "Red" are the two states in this model (in other words, you can be black, or you can be red).
Now create the model by following these steps:



1. **EMISSION STEP:** Roll a die. Note the number that comes up. This is the ***emission***. In the above graphic, I chose a red die to start (arbitrary — I could have chosen black) and rolled 2.
2. **TRANSITION STEP**: Randomly choose a ball from the bag with the color that matches the die you rolled in step 1. I rolled a red die, so I'm going to choose a ball from the red bag. I pulled out a black ball, so I'm going to transition to the black die for the next emission.

You can then repeat these steps to a certain number of emissions. For example, repeating this sequence of steps 10 times might give you the set {2,3,6,1,1,4,5,3,4,1}. **The process of transitioning from one *state* to the next is called a *Markov process*.**

Transitioning from red to black or black to red carries different probabilities as there are different numbers of black and red balls in the bags. The following diagram shows the probabilities for this particular model, which has two states (black and red):



PROBABILITY OF TRANSITIONING FROM ONE STATE TO THE NEXT

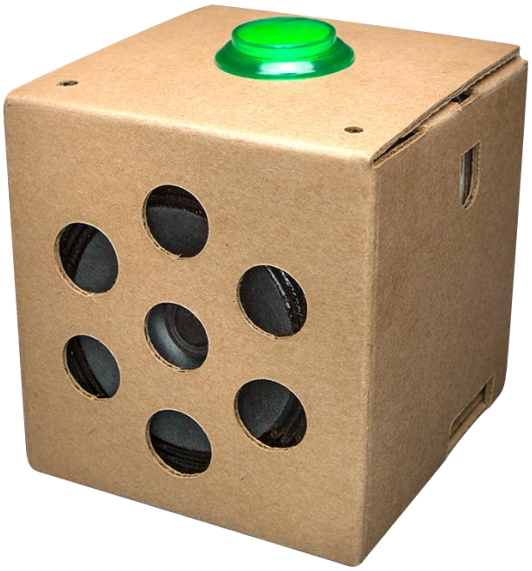
Neural Networks:

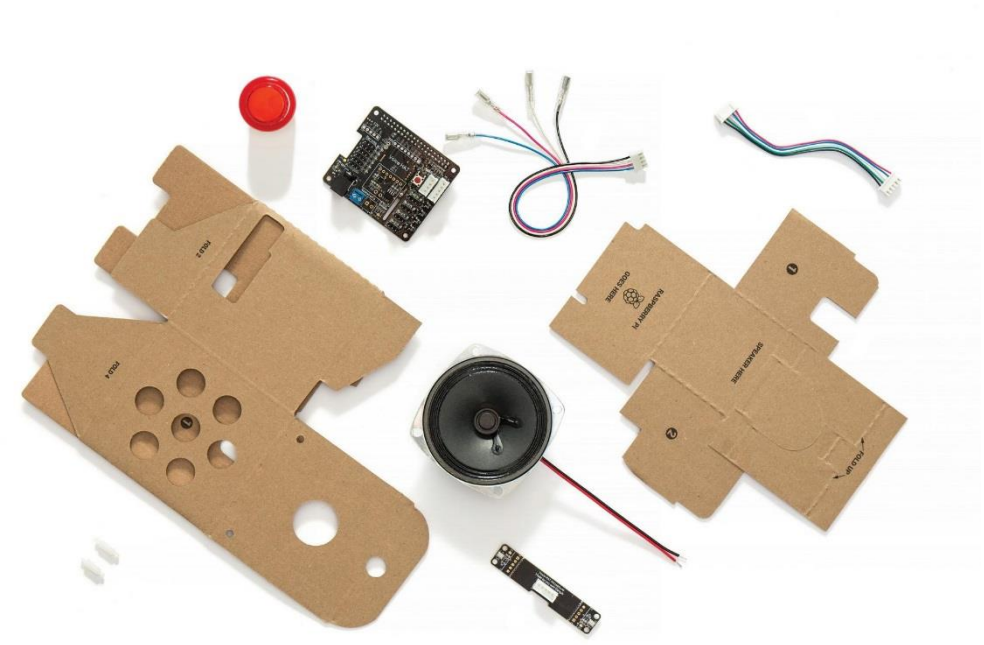
A neural network consists of many simple processing units (artificial neurons) each of which is connected to many other units. Each unit has a numerical activation level (analogous to the firing rate of real neurons). The only computation that an individual unit can do is to compute a new activation level based on the activations of the units it is connected to.

The connections between units are weighted and the new activation is usually calculated as a function of the sum of the weighted inputs from other units. Some units in a network are usually designated as input units which mean that their activations are set by the external environment. Other units are output units, their values are set by the activation within the network and they are read as the result of a computation. Those units which are neither input nor output units are called hidden units.

Speech Recognition Hardware used in our project

Voice Kit V1





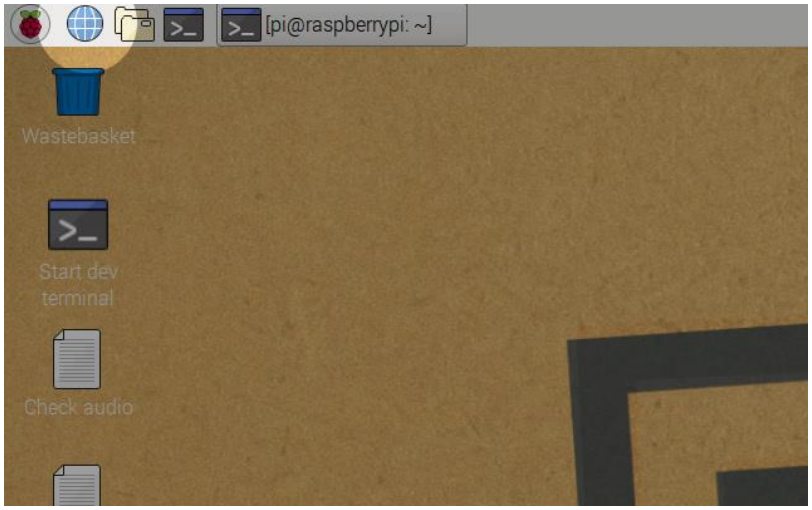
In the Kit:

- Voice HAT accessory board (×1)
- Voice HAT microphone board (×1)
- Plastic standoffs (×2)
- 3” speaker (wires attached) (×1)
- Arcade-style push button (×1)
- 4-wire button cable (×1)
- 5-wire daughter board cable (×1)
- External cardboard box (×1)
- Internal cardboard frame (×1)
- Raspberry Pi 3 (x1)
- SD card (x1)
- Size “00” Phillips screwdriver (x1)
- Scotch Tape (x1)

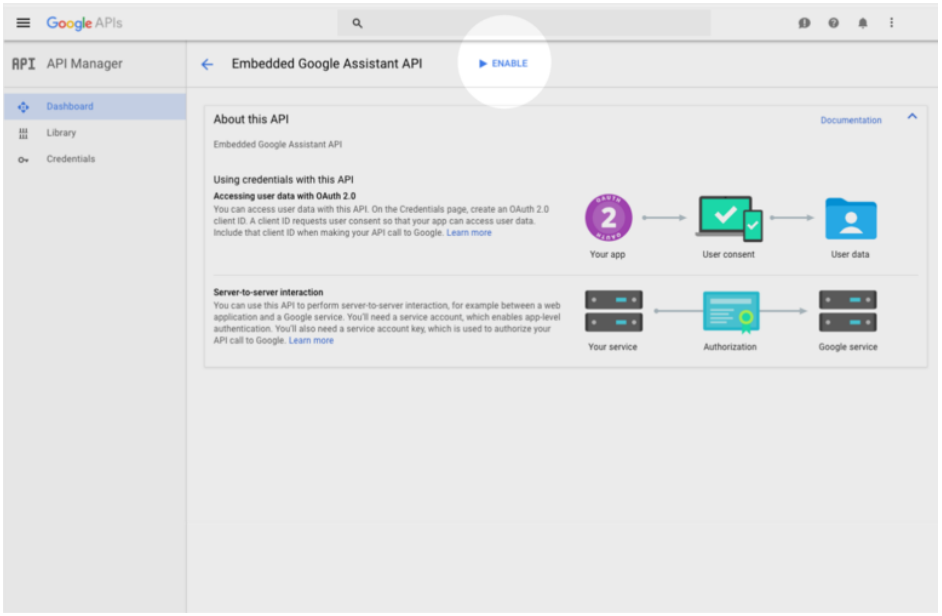
Steps to be followed:-

1. CONNECT TO GOOGLE CLOUD PLATFORM

To try the Google Assistant API, you need to first sign into Google Cloud Platform (GCP) and then enable the API.



2. TURN ON THE GOOGLE ASSISTANT API



Advantages of Speech Recognition:

Speech recognition technology also makes invaluable contributions to organizations. Businesses which provide customer services benefit from the technology in order to improve self-service in a way that enriches customer experience and reduces organizational costs. With the help of the voice recognition technology callers can input information such as name, account number, the reason of their call etc. without interacting with a live agent. Instead of having callers remain idly on hold while agents are busy, organizations can engage their callers without live customer representatives. That is why speech recognition technology contributes to cost savings by minimizing or even eliminating the need of live agents while improving customer experience.

Call centers which are continually challenged to balance customer satisfaction with cost containment apply voice recognition technology in order to benefit from invaluable advantages of the technology. Speech recognition technology;

- delivers a great customer experience while improving self-service system's containment rate
- encourages natural, human-like conversations that create more satisfying self-service interactions with customers
- automates what touchtone cannot by collecting dynamic data such as names and addresses
- enables organizations save agents for more important tasks

Conclusion:

This technology will spawn revolutionary changes in the modern world and become a pivot technology. Within five years, speech recognition technology will become so pervasive in our daily lives that service environments lacking this technology will be considered inferior.

APPENDIX:

OBJECT DETECTION:

Code Object Detection with SURF+FLANN

```
/**
 * It searches for on object inside scene using the SURF for keypoints and descriptors detection, and FLANN+KNN for matching them.
 * This implementation is done only in CPU.
 void processWithCpu(string objectInputFile, string sceneInputFile,    string outputFile, int minHessian = 100)
{
        // Load the image from the disk
        Mat img_object = imread( objectInputFile, IMREAD_GRAYSCALE ); // surf works only with grayscale images
        Mat img_scene = imread( sceneInputFile, IMREAD_GRAYSCALE );
    if( !img_object.data || !img_scene.data ) {
                std::cout<< "Error reading images." << std::endl;
                return;
        }
        // Start the timer
        GpuTimer timer;
        timer.Start();
        vector<KeyPoint> keypoints_object, keypoints_scene; // keypoints
        Mat descriptors_object, descriptors_scene; // descriptors (features)

        //-- Steps 1 + 2, detect the keypoints and compute descriptors, both in one method
        Ptr<SURF> surf = SURF::create( minHessian );
        surf->detectAndCompute( img_object, noArray(), keypoints_object, descriptors_object );
```

```
surf->detectAndCompute( img_scene, noArray(), keypoints_scene, descriptors_scene );

//-- Step 3: Matching descriptor vectors using FLANN matcher

FlannBasedMatcher matcher; // FLANN - Fast Library for Approximate Nearest Neighbors

vector< vector< DMatch> > matches;

matcher.knnMatch( descriptors_object, descriptors_scene, matches, 2 ); // find the best 2 matches of each descriptor

timer.Stop();

printf( "Method processImage() ran in: %f msecs.\n", timer.Elapsed() );

//-- Step 4: Select only good matches

std::vector< DMatch > good_matches;

for (int k = 0; k < std::min(descriptors_scene.rows - 1, (int)matches.size()); k++)

{

    if ( (matches[k][0].distance < 0.6*(matches[k][1].distance)) && ((int)matches[k].size() <= 2 && (int)matches[k].size()>0) )

    {// take the first result only if its distance is smaller than 0.6*second_best_dist

        // that means this descriptor is ignored if the second distance is bigger or of similar

    good_matches.push_back( matches[k][0] );

    }

}

//-- Step 5: Draw lines between the good matching points

Mat img_matches;

drawMatches( img_object, keypoints_object, img_scene, keypoints_scene,

        good_matches, img_matches, Scalar::all(-1), Scalar::all(-1),

        vector<char>(), DrawMatchesFlags::DEFAULT );

//-- Step 6: Localize the object inside the scene image with a square
```

```
        localizeInImage(good_matches,keypoints_object, keypoints_scene, img_object,
img_matches );

        //-- Step 7: Show/save matches

        //imshow("Good Matches & Object detection", img_matches);

        //waitKey(0);

        imwrite(outputFile, img_matches);

}
```

CODE FOR COLOUR TRACKING:

```
# A color code is a blob composed of two or more colors.
import sensor, image, time
from pyb import UART
from pyb import LED
blue_led  = LED(3)
green_led  = LED(2)
red_led  = LED(1)
uart = UART(3, 115200, timeout_char = 1000)
blue_led.on()
# Color Tracking Thresholds (L Min, L Max, A Min, A Max, B Min, B Max)
# The below thresholds track in general red/green things. You may wish to tune
them...
#thresholds = [(30, 100, 15, 127, 15, 127), # generic_red_thresholds -> index is 0 so code
== (1 << 0)
#           (30, 100, -64, -8, -32, 32)] # generic_green_thresholds -> index is 1 so code == (1
<< 1)
thresholds = [(55, 100,-24, 11, 32, 86),    #1#yellow
          (27, 100, 42, 80, 30, 64),    #2#red
          (39, 100,-51,-12, 10, 57)]    #4#green
# Codes are or'ed together when "merge=True" for "find_blobs".
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)
sensor.set_auto_gain(False) # must be turned off for color tracking
sensor.set_auto_whitebal(False) # must be turned off for color tracking
clock = time.clock()
blue_led.off()
green_led.off()
red_led.off()
object_x_old = 0
```

```
object_y_old = 0
code = 2 ## 1:yellow   2:red    4:green
buf = "00"
# Only blobs that with more pixels than "pixel_threshold" and more area than
"area_threshold" are
# returned by "find_blobs" below. Change "pixels_threshold" and "area_threshold" if
you change the
# camera resolution. "merge=True" must be set to merge overlapping color blobs for
color codes.
while(True):
   clock.tick()
   blue_led.off()
   green_led.off()
   red_led.off()
   img = sensor.snapshot()
   for blob in img.find_blobs(thresholds, pixels_threshold=100, area_threshold=100,
merge=False):
#check with color should be detect
      if uart.any()>0 :
         buf=uart.read()
         print (buf[0])
         if buf[0]==ord('y') :
            code = 1
         if buf[0]==ord('r') :
            code = 2
         if buf[0]==ord('g') :
            code = 4
#check if there is object with right color
      if blob.code() == code:
         img.draw_rectangle(blob.rect())
         img.draw_cross(blob.cx(), blob.cy())
         #print(blob.cx(), blob.cy(),blob.w())
#make sure the detected object is stable and print the coordinates
#first it detect if the coordinates of blob is available
#second compared with the last position to make sure if the object is not moving
#third reduce the affect of anbience
         if blob.cx()!=None and (
            abs(object_x_old - int(blob.cx())) < 8 and
            abs(object_y_old - int(blob.cy())) < 8) and (
            blob.w()>35 and
            blob.h()>35):
           #just detect the objects. turn on the blue only
```

```
            blue_led.on()
            red_led.off()
            green_led.off()
            #print("stable!")
            #print (buf)
#check if the uart got any command and response
            #if uart.any()>0 :
                #buf=uart.read(1)
            if buf[1]==ord('S') :
                #print("command\n")
                #detect both the objects and the vision command from mega2560. turn on
the red only
                blue_led.off()
                red_led.on()
                green_led.off()
                uart.write('x'+str(blob.cx())+'y'+str(blob.cy())+'\n')
                #finish the sending. turn on the green only
                blue_led.off()
                red_led.off()
                green_led.on()
                #clear the flag
                buf = "00"
            object_x_old = int(blob.cx())
            object_y_old = int(blob.cy())
```

GESTURE CONTROL:

SENDER CIRCUIT ARDUINO CODE:

```
//to send accelerometer data via nodemcu
#include <Wire.h>
#include <SoftwareSerial.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <SerialESP8266wifi.h>
#include <SPI.h>

const int MPU_addr=0x68;  // I2C address of the MPU-6050
int16_t x,y,z;
int16_t AcX,AcY,AcZ;
byte ledPin = 2;
char ssid[] = "xyz";            // SSID of your home WiFi
char pass[] = "12345678";           // password of your home WiFi
WiFiServer server(80);

IPAddress ip(192, 168, 0, 80);          // IP address of the server
IPAddress gateway(192,168,0,1);         // gateway of your network
IPAddress subnet(255,255,255,0);        // subnet mask of your network
void setup() {
  Wire.begin(); // initiate i2c system
  Wire.beginTransmission(MPU_addr); // be sure we talk to our MPU vs some
other device
```

```
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true); // done talking over to MPU device, for the moment
  Serial.begin(115200);               // only for debug
  WiFi.config(ip, gateway, subnet);      // forces to use the fix IP
  WiFi.begin(ssid, pass);              // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  server.begin();                     // starts the server
  Serial.println("Connected to wifi");
  Serial.print("Status: ");
  Serial.println(WiFi.status());
  pinMode(ledPin, OUTPUT);

}
void loop () {
  Wire.beginTransmission(MPU_addr); // get ready to talk to MPU again
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false); // done talking to MPU for the time being
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  // all the fancy <<8| stuff is to bit shift the first 8 bits to
  // the left & combine it with the next 8 bits to form 16 bits
  AcX=Wire.read()<<8|Wire.read();     // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
```

```
  AcY=Wire.read()<<8|Wire.read();     // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();     // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  // the above lines have gathered Accellerometer values for X, Y, Z
  int x = map(AcX, -17000, 17000, 0, 179);
  int y = map(AcY, -17000, 17000, 0, 179);
  int z = map(AcZ, -17000, 17000, 0, 179);
  WiFiClient client = server.available();
  if (client) {
    if (client.connected()) {
      digitalWrite(ledPin, LOW);  // to show the communication only (inverted logic)
      Serial.println(".");
      client.println(x);
      client.println(y);
      client.println(z);
      client.flush();
      digitalWrite(ledPin, HIGH);
      delay(200);
    }
  }
}
```

RECEIVER CIRCUIT ARDUINO CODE:

```
//code to receive and instruct motors
#include <Wire.h>
#include <SoftwareSerial.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <SPI.h>
#include <SerialESP8266wifi.h>
#include <Servo.h>

Servo servo1;
Servo servo2;
Servo servo3;

int x,y,z;

byte ledPin = 2;
char ssid[] = "xyz";          // SSID of your home WiFi
char pass[] = "12345678"; // password of your home WiFi

unsigned long askTimer = 0;

IPAddress server(192,168,0,80);       // the fix IP address of the server
WiFiClient client;
```

```
void setup() {
  Wire.begin();
  Serial.begin(115200);            // setup of baud rate and serial communication
  WiFi.begin(ssid, pass);           // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print("."); // prints '.' until connected to WiFi router
    delay(500);
  }
  Serial.println("Connected to wifi");
  Serial.print("Status: ");
  Serial.println(WiFi.status());
  pinMode(ledPin, OUTPUT);
  servo1.attach(8);
  servo2.attach(9);
  servo3.attach(10);
}
void loop () {
  client.connect(server, 80);   // Connection to the server
  digitalWrite(ledPin, LOW);     // to show the communication only (inverted
logic)
  Serial.println(".");
  Serial.println();
  int x = client.read();
  int y = client.read();
  int z = client.read();
  Serial.print("X = " + x);
  Serial.print(" Y = " + y);
```

```
  Serial.print(" Z = " + z);
  Serial.println();
  client.flush();
  digitalWrite(ledPin, HIGH);
  delay(200);  // client will trigger the communication after one second(s)
  servo1.write(x);
  servo2.write(y);
  servo3.write(z);
  delay(50);
}
```

FLEX SENSOR ARDUINO CODE:

```
#include <Servo.h>
const int pinflex = 0;
Servo myServo;
void setup() {
myServo.attach(10);
}
void loop() {
int flexpos;
int servopos;
flexpos = analogRead(pinflex);
servopos = map(flexpos, 600, 900, 0, 180);
servopos = constrain(servopos, 0, 180);
myServo.write(servopos);
delay(25);
}
```

MATLAB CODE FOR INVERSE KINEMATIC ANALYSIS:

```
% THIS PROGRAM IS USED TO SOLVE THE INVERSE KINEMATIC OF THE ARM
% DEFINE A NON RETURN FUNCTION TO COMBINE ALL THE INVERSE FUNCTIONS TOGETHER
% IN ONE SCRIPT
function [ NONRETURNFUNCTION] = INVERSE( )
% DECLERATION OF THE ROBOT PARAMETER
d1 = 352;
a1 = 70;
a2 = 360;
d4 = 380;
NOSOLUTION=1000;
d6 = 65;
% USER INTERFACE
xtip = input ('ENTER THE GOAL POSTION X = ');
ytip = input ('ENTER THE GOAL POSTION y = ');
ztip = input ('ENTER THE GOAL POSTION z = ');
alpha= input ('ENTER THE VALUE OF alpha IN DEGREE = ');
beta = input ('ENTER THE VALUE OF beta IN DEGREE = ');
gama = input ('ENTER THE VALUE OF gama IN DEGREE = ');
% CALCULATING ALL THE POSSIBLE VALUES FOR THETA1
theta1= atan2 (ytip,xtip);
theta11= pi + theta1;
THETA1 = theta1 * 180/pi;
THETA11= theta11 * 180/pi;
% CALCULATING ALL THE POSSIBLE VALUES FOR THETA3
s = (ztip - d1);
r = sqrt((xtip - a1*cos (theta1))^2 +(ytip - a1*sin(theta1))^2);
czeta = (r^2 + s^2 - (a2)^2 - (d4 + d6)^2)/(2 * a2 *(d4 + d6));
% SINGULARTIY CONDTION, CHECK IF THE POSTION WITHIN THE WORKSPACE OR NOT
if (abs(czeta) <= 1)
szeta = sqrt(1-(czeta)^2);
szeta1 = -szeta;
zeta= atan2(szeta,czeta);
zeta1= atan2(szeta1,czeta);
theta3 = -(pi/2 + zeta);
theta33 = -(pi/2 + zeta1);
THETA3 = conversion( theta3,50,-230);
THETA33 = conversion( theta33,50,-230);
else
theta3 = NOSOLUTION;
theta33= NOSOLUTION;
THETA3 = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA3');
THETA33 = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA33');
End
s = (ztip - d1);
r = sqrt((xtip - a1*cos (theta11))^2 +(ytip - a1*sin(theta11))^2);
czetai = (r^2 + s^2 - (a2)^2 - (d4 + d6)^2)/(2 * a2 *(d4 + d6));
```

```
% SINGULARTIY CONDTION, CHECK IF THE POSTION WITHIN THE WORKSPACE OR NOT
if (abs(czetai) <= 1)
szetai = sqrt(1-(czetai)^2);
szeta1i = -szetai;
zetai= atan2(szetai,czetai);
zeta1i= atan2(szeta1i,czetai);
theta3i = -(pi/2 + zetai);
theta33i = -(pi/2 + zeta1i);
THETA3i = conversion( theta3i,50,-230);
THETA33i = conversion( theta33i,50,-230);
else
theta3i=NOSOLUTION;
theta33i=NOSOLUTION;
THETA3i = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA3i');
THETA33i = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA33i');
end
% CALCULATING ALL THE POSSIBLE VALUES FOR THETA2
if (theta3 == NOSOLUTION)
THETA2 = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA2');
THETA22 = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA22');
else
theta2 = THE2(xtip,ytip,ztip,theta1,zeta);
theta22 = THE2COMP(xtip,ytip,ztip,theta1,zeta);
THETA2 = conversion( theta2,110,-90);
THETA22 = conversion( theta22,110,-90);
end
if (theta33 == NOSOLUTION)
THETA2i = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA2i');
THETA22i = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA22i');
else
theta2i = THE2(xtip,ytip,ztip,theta1,zeta1);
theta22i = THE2COMP(xtip,ytip,ztip,theta1,zeta1);
THETA2i = conversion( theta2i,110,-90);
```

```
THETA22i = conversion( theta22i,110,-90);
end
if (theta3i == NOSOLUTION)
THETA2j = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA2j');
THETA22j = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA22j');
else
theta2j = THE2(xtip,ytip,ztip,theta11,zetai);
theta22j = THE2COMP(xtip,ytip,ztip,theta11,zetai);
THETA2j = conversion( theta2j,100,-90);
THETA22j = conversion( theta22j,100,-90);
end
if (theta33i == NOSOLUTION)
THETA2k = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA2k');
THETA22k = ('GOAL OUT OF WORKSPACE, THERE IS NO VAILD VALUS FOR THETA22k');
else
theta2k = THE2(xtip,ytip,ztip,theta11,zeta1i);
theta22k = THE2COMP(xtip,ytip,ztip,theta11,zeta1i);
THETA2k = conversion( theta2k,110,-90);
THETA22k = conversion( theta22k,110,-90);
end
```

```
% DISPLAY ALL THE POSSIBLE EIGHT SOLUTIONS, NOTE THAT EVERY TWO SOLUTIONS FORM
ONLY ONE SOLUTION SET
disp ( ' THETA 1,2,3 SOLUTIONS')
disp ( ' SET 1')
SOL1 =[ THETA1, THETA2, THETA3]
SOL2 =[ THETA1, THETA22, THETA3]
disp ( ' SET 2')
SOL3 =[ THETA1, THETA2i, THETA33]
SOL4 =[ THETA1, THETA22i, THETA33]
disp ( ' SET 3')
SOL5 =[ THETA11, THETA2j, THETA3i]
SOL6 =[ THETA11, THETA22j, THETA3i]
disp ( ' SET 4')
SOL7 =[ THETA11, THETA2k, THETA33i]
SOL8 =[ THETA11, THETA22k, THETA33i]
% SOLVING THE SECOND KINEMATIC SUB-PROBLEM (ORIENTATION)
alpha = alpha * pi/180;
beta = beta * pi/180;
gama = gama * pi/180;

R60 = [cos(alpha).*cos(beta),  (cos(alpha).*sin(beta).*sin(gama))- sin(alpha).*cos(gama),     (cos(alpha).*sin(beta).*cos(gama)) +sin(alpha).*sin(gama) ;
sin(alpha).*cos(beta),         (sin(alpha).*sin(beta).*sin(gama)) + cos(alpha).*cos(gama),    (sin(alpha).*sin(beta).*cos(gama)) - cos(alpha).*sin(gama) ;
- sin (beta),                  cos (beta).*sin (gama),                                            cos (beta).*sin (gama)]

R30 = [cos(theta1).*cos(theta2+theta3),    -cos(theta1).*sin(theta2+theta3),    sin(theta1);
sin (theta1).*cos(theta2+theta3),          -sin(theta1).*sin(theta2+theta3),    cos(theta1);
-sin(theta2+theta3),                       -cos(theta2+theta3),                       0 ] ;
RT30= transpose (R30);
R63 = RT30 * R60 ;
g11 = R63 (1,1);
g12 = R63 (1,2);
g23 = R63 (2,3);
g31 = R63 (3,1);
```

```
g32 = R63 (3,2);
g33 = R63 (3,3);
% THETA 4,5,6 CALCULATION
theta5 = atan2 ( sqrt((g31)^2 +(g32)^2), g33);
if(theta5 == 0)
THETA4= 0
THETA5= 0
theta6 = atan2 (-g12, g11);
THETA6= theta6*180/pi
elseif (theta5 == pi)
THETA4= 0
THETA5= 0
theta6 = atan2 (g12,-g11);
THETA6= theta6*180/pi
else
theta4 = atan2 (g32/ sin (theta5), - g31/ sin (theta5));
theta6 = atan2 (g23/ sin (theta5), g31/ sin (theta5));
THETA4= conversion( theta4,200,-200);
THETA5= conversion( theta5,115,-115);
THETA6= conversion( theta6,400,-400);
% FLIPPED POSTION
theta44 = theta4 + pi;
theta55 = -theta5;
theta66 = theta6+pi;
THETA44= conversion( theta44,200,-200);
THETA55= conversion( theta55,115,-115);
THETA66= conversion( theta66,400,-400);
disp ( ' THETA 4,5,6 SOLUTIONS')
Solution1 = [THETA4,THETA5,THETA6]
Solution2 = [THETA44,THETA55,THETA66]
end
```

```
% FIRST POSSIBLE SOLUTION OF THETA2 FUNCTION
function RES = THE2(xtip,ytip,ztip,theta1,zeta)
s = (ztip - d1);
r = sqrt((xtip - a1*cos (theta1))^2 +(ytip - a1*sin(theta1))^2);
omega = atan2 (s, r);
lenda = atan2 (( d4+d6) * sin (zeta) , a2+( d4+d6)* cos (zeta));
RES = - ((omega - lenda)- ( pi/2)) ;
End
% SECOND POSSIBLE SOLUTION OF THETA2 FUNCTION
function RES1 = THE2COMP(xtip,ytip,ztip,theta1,zeta)
s = (ztip - d1);
r = - sqrt((xtip - a1*cos (theta1))^2 +(ytip - a1*sin(theta1))^2);
omega = atan2 (s, r);
lenda = atan2 (( d4+d6) * sin (zeta) , a2+( d4+d6)* cos (zeta));
RES1 = - ((omega - lenda) - ( pi/2));
end
% JOINT ANGLES LIMIT FUNCTION
function OUT = conversion( theta,upperlimit,lowerlimit)
upperlimit = upperlimit * pi / 180;
lowerlimit = lowerlimit * pi / 180;
if (theta > upperlimit)
OUT = (' THE SOLUTION OUT OF JOINT ANGLE LIMIT ');
elseif (theta < lowerlimit)
OUT = (' THE SOLUTION OUT OF JOINT ANGLE LIMIT ');
else
OUT = theta * 180 / pi;
end
end
end
```

REFERENCES

For design and analysis:

Books

Narayana K. L., Kannaiah P., Venkata Reddy K., “Machine Drawing”, 3rd Edition, New Age International Publishers, New Delhi.

Uicker J.J., Pennock G.R., Shigley J.E., “Theory of Machines and Mechanisms”, Oxford Univ. Press, NY, 3rd Ed., 2003

The Finite Element Method for Engineers, 4ed by Kenneth H. Huebner (Author), Donald L. Dewhirst (Author), Doughlas E. Smith (Author),2008

Articles

https://www.researchgate.net/publication/317409806_Forward_and_Inverse_Kinematic_Analysis_and_Validation_of_the_ABB_IRB_140_Industrial_Robot

http://www.mecs-press.org/ijisa/ijisa-v6-n4/IJISA-V6-N4-3.pdf

Websites

https://en.wikipedia.org/wiki/Finite_element_method

nptel.ac.in/courses/112104116/

For Object Detection:

Object Detection Techniques Applied on Mobile Robot Semantic Navigation- http://www.mdpi.com/1424-8220/14/4/6734/pdf

The Open Source Computer Vision (OpenCV). Available online: http://opencv.org

Khan, N.Y.; McCane, B.; Wyvill, G. SIFT and SURF performance evaluation against various image deformations on benchmark dataset. In Proceedings of the 2011 International Conference on Digital Image Computing Techniques and Applications (DICTA), Queensland, Australia, 6–8 December 2011; pp. 501–506.

Bay, H.; Tuytelaars, T.; van Gool, L. SURF: Sped up robust features. In Computer Vision—ECCV 2006; Springer: Berlin/Heidelberg, Germany, 2006.

Muja, M.; Lowe, D. FLANN—Fast Library for Approximate Nearest Neighbors User Manual; Computer Science Department, University of British Columbia, Vancouver, BC, Canada, 2009.

http://docs.openmv.io/openmvcam/tutorial/index.html

https://3sidedcube.com/guide-retraining-object-detection-models-tensorflow/

http://jultika.oulu.fi/files/nbnfioulu-201802081173.pdf

http://www.coldvision.io/2016/06/27/object-detection-surf-knn-flann-opencv-3-x-cuda/

http://forums.openmv.io/viewforum.php?f=6&sid=890ba6bf96c0b943421df22435538269

https://openmv.io/blogs/news

For Gesture Control:

https://tttapa.github.io/ESP8266/Chap03%20-%20Software.html

http://www.instructables.com/id/MPU6050-Controlled-Servo-Arm/

http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/

https://maker.pro/arduino/tutorial/how-to-control-a-servo-with-an-arduino-and-mpu6050

http://icircuit.net/interfacing-arduino-uno-wtih-nodemcu/1474

http://www.electronicwings.com/nodemcu/nodemcu-spi-with-arduino-ide

https://www.brainy-bits.com/how-to-use-the-nrf24l01-2-4ghz-wireless-module/

https://dominicm.com/esp8266-send-receive-data/

http://randomnerdtutorials.com/esp8266-web-server-with-arduino-ide/

http://www.instructables.com/id/IoT-ESP8266-Series-1-Connect-to-WIFI-Router/

http://fab.cba.mit.edu/classes/865.15/people/dan.chen/esp8266/

https://arduino.stackexchange.com/questions/18575/send-at-commands-to-esp8266-from-arduino-uno-via-a-softwareserial-port

http://www.instructables.com/id/WiFi-Communication-Between-Two-ESP8266-Based-MCU-T/

http://www.instructables.com/id/Accelerometer-controlled-robot-pickup-and-place/

GLOSSARY

ACCELEROMETER:

> An instrument for measuring the acceleration of a moving or vibrating body.

DATA DECODER:

> A device capable of converting audio or video signals into a different form, for example from digital to analogue.

DATA ENCODER:

> An encoder is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another, for the purposes of standardization, speed or compression.

ESP8266:

The ESP8266 WiFi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network.

GYROSCOPE:

A device consisting of a wheel or disc mounted so that it can spin rapidly about an axis which is itself free to alter in direction.

GPIO:

It refers to General Purpose Input Output pins in a circuit.

I2C PROTOCOL:

The Inter-integrated Circuit (I^2C) Protocol is a protocol intended to allow multiple “slave” digital integrated circuits (“chips”) to communicate with one or more “master” chips.

IDE:

An integrated development environment (IDE) is a software suite that consolidates the basic tools developers need to write and test software.

MAGNETOMETER:

An instrument used for measuring magnetic forces, especially the earth's magnetism.

SPI:

Serial peripheral interface (SPI) is an interface that enables the serial (one bit at a time) exchange of data between two devices, one called a master and the other called a slave.