



DESIGN AND FABRICATION OF ARDUINO BASED ROBOTIC ARM

J. Sai Dilip¹, K.Srinivasrao¹, M. Sai Anudeep¹, D.Laxman Shyam¹, M. Manohar¹, Dr. S. Ramana Babu Sir (Asst Professor)²

¹Final Year B. Tech Students, Department of Mechanical Engineering, Sanketika Vidya Parishad Engineering College, P.M Palem, Visakhapatnam, 530041, India.

²Assistant Professor, Department of Mechanical Engineering, Sanketika Vidya Parishad Engineering College, P.M Palem, Visakhapatnam, 530041, India.

ABSTRACT

In Order To Make The Human Life Easier And More Efficient We Need A Machine Which Has The Potential To Do The Work As Humans Does . Therefore We Build An Robotic Arm Which Runs By A Computer Programme . A Mechanical Arm Is A Machine That Mimics The Action Of A Human Arm. Mechanical Arms Are Composed Of Multiple Beams Connected By Hinges Powered By Actuators. One End Of The Arm Is Attached To A Firm Base While The Other Has A Tool. They Can Be Controlled By Humans Either Directly Or Over A Distance. A Computer-Controlled Mechanical Arm Is Called A Robotic Arm.

Here We Propose to Build a Robotic Arm controlled By Natural Human Arm Movements Whose Data is Acquired Through The Use Of Accelerometers Or Without it Through Manual Control. The Development Of This Arm Is based On Atmega32 And Atmega640 Platform Through arduino UNO Or MEGA Along With A Personal Computer For signal Processing, Which Will All Be Interfaced With Each other Using Serial Communication. Finally,

This Proto type of The Arm May Be Expected to Overcome the Problem such As Placing Or Picking Hazardous Objects Or Non-Hazardous Objects That Are Far Away From The User And Also is Used Where Displacement Of Very Heavy Objects Is needed From One Place To Another Or Automation Required in Many Industries.

Keywords: *mechanical arm, Actuators, Accelerometers, Atmega640, Arduino Uno .*

1. INTRODUCTION

From manufacturing to automotive to agriculture, industrial robotic arms are one of the most common types of robots in use today.

Robotic arms, also known as articulated robotic arms, are fast, reliable, and accurate and can be programmed to do an infinite number of tasks in a variety of environments. They are used in factories to automate execution of repetitive tasks, such as applying paint to equipment or parts; in warehouses to pick, select, or sort goods from distribution conveyors to fulfill consumer orders; or in a farm field to pick and place ripe fruits onto storage trays. And as robotic technologies develop and industrial environments become more connected, the capabilities of robotic arms expand to enable new use cases and business operation models.

In the past, a robotic arm required teaching to perform narrowly defined tasks, such as picking a single type of object from a precise location with a specific orientation. Robots were not able to identify a particular type of object among many, determine an object location with some tolerance (area rather than exact position), or adjust the grasp based on object orientation.

1.1 TYPES OF ROBOTIC ARM'S

- Cartesian robot / Gantry robot: Used for pick and place work, application of sealant, assembly operations, handling machine tools and arc welding. It is a robot whose arm has three prismatic joints, whose axes are coincident with a Cartesian coordinator.
- Collaborative robot / Cobot: Cobot applications contrast with traditional industrial robot applications in which robots are isolated from human contact. Cobot safety may rely on lightweight construction materials, rounded edges, and the inherent limitation of speed and force, or on sensors and software that ensures safe behavior.

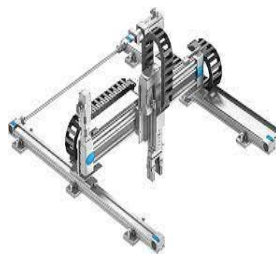
- Cylindrical robot: Used for assembly operations, handling at machine tools, spot welding, and handling at die casting machines. It is a robot whose axes form a cylindrical coordinate system.
- Spherical robot / Polar robot: Used for handling machine tools, spot welding, die casting, fettling machines, gas welding and arc welding. It is a robot whose axes form a polar coordinate system.
- SCARA 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.
- Articulated robot: Used for assembly operations, die casting, fettling machines, gas welding, arc welding and spray painting. It is a robot whose arm has at least three rotary joints.
- Parallel robot: One use is a mobile platform handling cockpit flight simulators. It is a robot whose arms have concurrent prismatic or rotary joints.
- Anthropomorphic robot: It is shaped in a way that resembles a human hand, i.e. with independent fingers and thumbs.

1.2 DIFFERENT TYPES OF ROBOTIC ARM'S

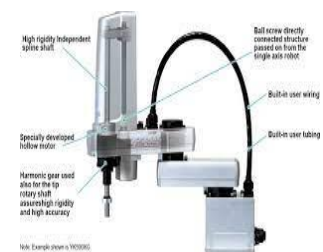
Articulated Robot Arm



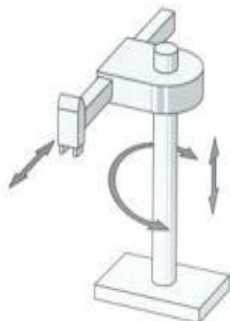
Cartesian or Rectangular Robots



SCARA Robots



Polar or Spherical Robots



Delta or Parallel Robots



Collaborative Robots/ Cobots



1.3 SIX-AXIS ROBOTS

One of the most popular robot types in the industrial space is the six-axis articulated-arm robot. Six axes allow a robot to move in the x, y, and z planes, advantages of six-axis robots include mobility (easy to move and/or mount) and a wide horizontal and vertical reach.

In primary and secondary packaging operations, six-axis robots are most commonly used for case and tray packing, cartoning, depalletizing and palletizing, and even truck loading.

The robot's design includes a new arm structure and drive unit and a lighter main to unit to enable high speeds

The robot's internal Ethernet wiring is said to enable easy connection to a vision system or other peripheral devices, and its floor, ceiling, and wall-mount options provide installation flexibility for almost any layout.

2. COMPONENTS

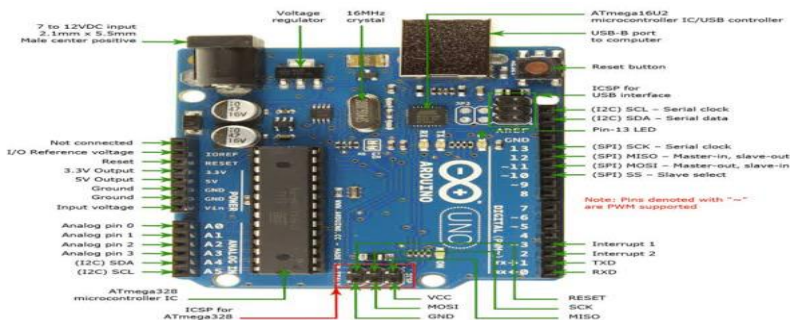
1. Arduino Uno
2. 1000uF Capacitor (4 pieces)
3. 100nF Capacitor (4 pieces)
4. Servo Motor (SG 90- four pieces)
5. 10K pot- Variable Resistor (4 pieces)
6. Power Supply (5v- preferably two)
7. Bluetooth with android application

Arduino nano microcontroller

Although microcontroller type PIC is usually used in programming and software field, Arduino has become very popular in the world in recent times. It is based on Arduino's past wiring and processing projects. Processing is written for non-programming users. Arduino wiring is produced on the basis of the programming language. The common feature of both is that it provides an environment where even the basic knowledge of electronics and programming can easily design. Arduino is now becoming more and more common nowadays. Even unmanned aerial vehicles made with Arduino, which is used almost every field, are visible. The causes of the spread of Arduino at such a rapid rate are; □ It can be used on all platforms due to the simplicity of the development environment with driver usage.

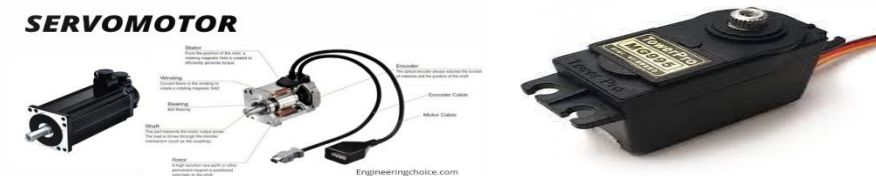
Page | 4

- With the help of the advanced library, even complex operations can be easily solved.
- Programs written in Arduino can run fast because they are not run on any other platform.
- There is a lot of hardware support that is compatible with Arduino and can work together.
- Communication with the environment is easy because it is open source.
- If there are any problems due to a large number of Arduino users, the solution can be easily reached.



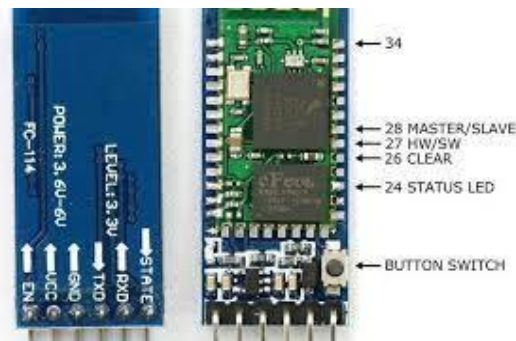
Servo motor

Servomotor Is a Rotary Actuator or Linear Actuator That Allows for Precise Control of Angular or Linear Position, Velocity and Acceleration. It Consists of a Suitable Motor Coupled to a Sensor for Position Feedback. A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analog or digital) representing the position commanded for the output shaft. The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller.

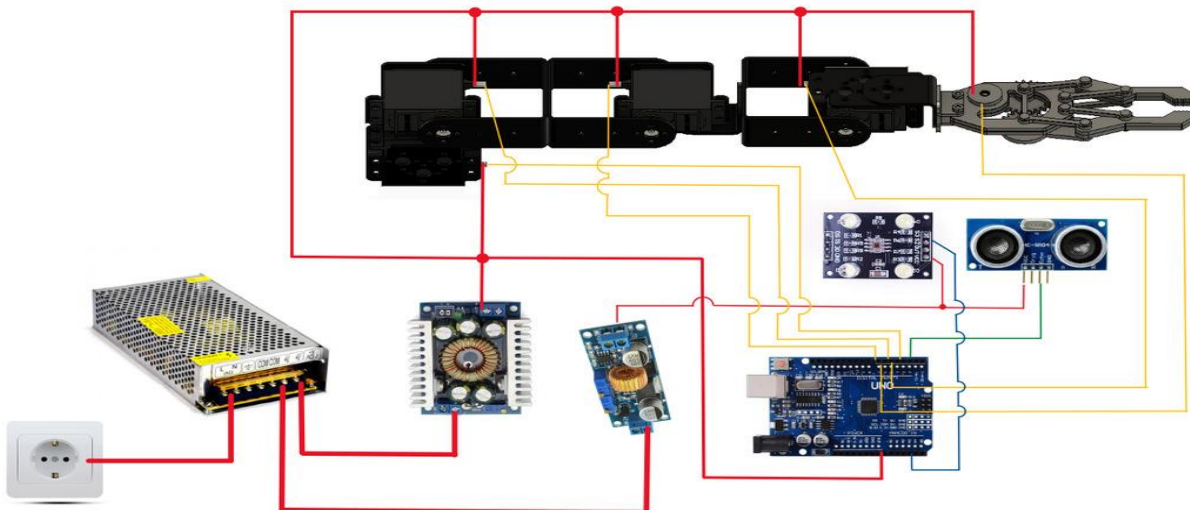


Bluetooth Module

It is a one of the great example for wireless connectivity. It is used in many fields. Bluetooth consumes very small amount of energy. Do you know about Smartphone controlled robot or car. Commonly one of these two wireless technology is used in Smartphone controlled robot. One is WIFI and other is Bluetooth.



Schematic line diagram of the circuit



Arm axis rotation

- **Axis 1:** this axis, located at the robot base, allows the robot to rotate from left to right. This sweeping motion extends the work area to include the area on either side and behind the arm. This axis allows the robot to spin up to a full 180 degree range from the center point. This axis is also known as the Motoman: S and Fanuc: J1.
- **Axis 2:** this axis allows the lower arm of the robot to extend forward and backward. It is the axis powering the movement of the entire lower arm. This axis is also known as the Motoman: L and Fanuc: J2.
- **Axis 3:** the axis extends the robot's vertical reach. It allows the upper arm to raise and lower. On some articulated models, it allows the upper arm to reach behind the body, further expanding the work envelope. This axis gives the upper arm the better part access. This axis is also known as the Motoman: U and Fanuc: J3.
- **Axis 4:** Working in conjunction with the axis 5, this axis aids in the positioning of the end effector and manipulation of the part. Known as the wrist roll, it rotates the upper arm in a circular motion moving parts between horizontal to vertical orientations. This axis is also known as the Motoman: R and Fanuc: J4.
- **Axis 5:** this axis allows the wrist of the robot arm to tilt up and down. This axis is responsible for the pitch and yaw motion. The pitch, or bend, motion is up and down, much like opening and closing a box lid. Yaw moves left and right, like a door on hinges. This axis is also known as the Motoman: B and Fanuc: J5.

- **Axis 6:** this is the wrist of the robot arm. It is responsible for a twisting motion, allowing it to rotate freely in a circular motion, both to position end effectors and to manipulate parts. It is usually capable of more than a 360 degree rotation in either a clockwise or counterclockwise direction. This axis is also known as the Motoman: T and Fanuc: J6.

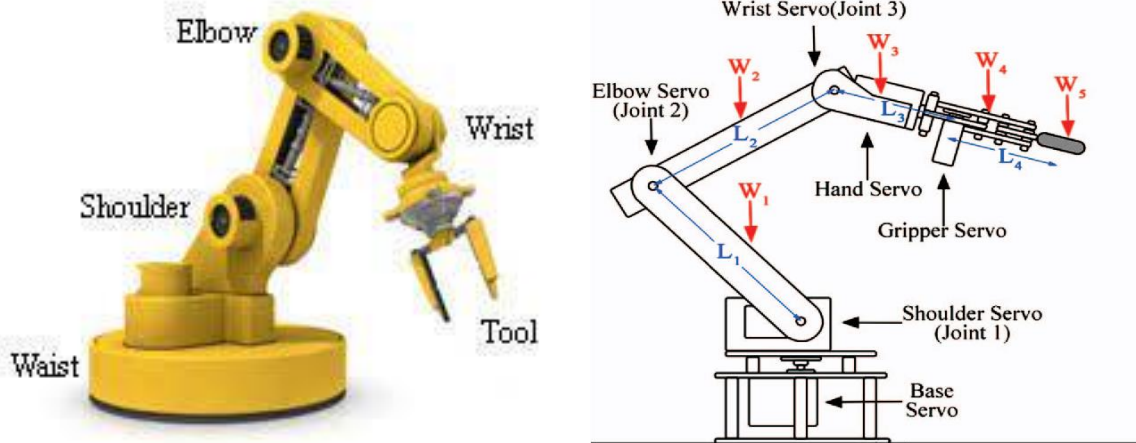
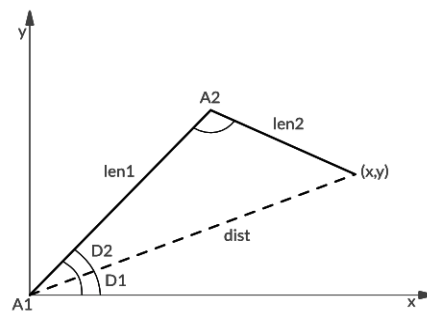


Fig. 1: Shape of the Robotic Arm

Working of the robotic arm

The basic function of a ROBOTIC ARM is done by its joints. Joints are analogous to human joints and are used to join the two consecutive rigid bodies in the robot. They can be rotary joint or linear joint. To add a joint to any link of a robot, we need to know about the degrees of freedom and degrees of movement for that body part. Degrees of freedom implement the linear and rotational movement of the body and Degrees of movement imply the number of axis the body can move.

3. APPLYING THE GEOMETRIC APPROACH TO THE SCARA ROBOT

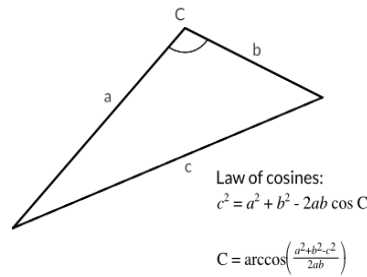


This diagram tells us a couple of things:

- The segments have the length $len1$ and $len2$, respectively.
- The root joint describes an angle $A1$ measured from the x axis.
- The second joint describes an angle $A2$ measured from the first segment (counterclockwise in both cases).
- The tip of segment 2 points to (x,y) , and we want to calculate back from that point to the yet unknown values of $A1$ and $A2$.

In the diagram you also see a new dotted line named $dist$. It points from $(0,0)$ to (x,y) , and as you can easily see, the three lines $dist$, $len1$, and $len2$ define a triangle. Furthermore, $dist$ divides angle $A1$ into two angles $D1$ and $D2$.

Now is a good moment to dig out an old trig formula you may remember from school: The law of cosines.

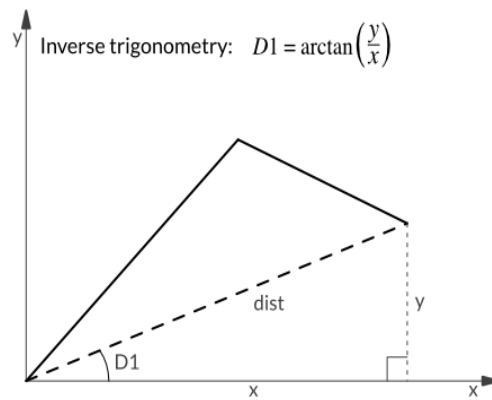


The law of cosines (see the first formula in the figure above) is a generalization of the Pythagorean theorem ($c^2 = a^2 + b^2$ for right(-angled) triangles) to arbitrary triangles. We do not need the basic form, but rather the transformed version that you can see below the original formula. With this version, we can calculate angle C from the triangle's sides a , b , and c . This comes handy in two places, as we'll see shortly.

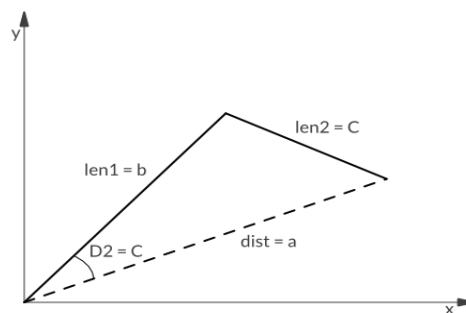
So how do we calculate $A1$ and $A2$?

- From the robotic arm diagram above (the one with $D1$, $D2$, $dist$, etc), we can directly derive the first formula:
- $A1 = D1 + D2$
- $D1$ is fairly easy to calculate. In the following diagram, x , y , and $dist$ define a right-angled triangle. Here, $D1$ can be calculated in two ways: The arcsine of $y/dist$ or the arctangent of y/x . As we have x and y readily available, let's choose the arctangent formula.

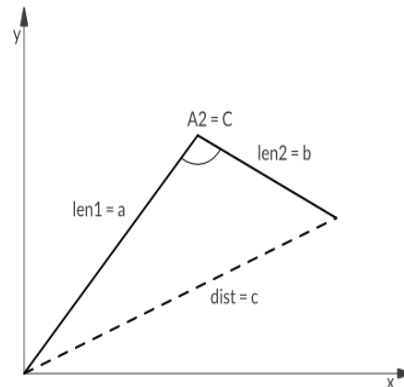
However, there is a problem hidden here. The formulas certainly are correct if x and y are positive, but what if either of the two, or even both, are negative? Luckily, a solution is available in form of a method from the standard library: `math.Atan2(y, x)`. It delivers the correct result for all possible combinations of x and y .



$D2$ requires the law of cosines. Basically, we just map our "robot triangle" to the "law of cosines" triangle by using $dist$ as a , $len1$ as b , and $len2$ as c . The resulting angle C is our $D2$.



Now only A_2 is left. Luckily, we can reuse the law of cosines for this. We only need to map our triangle to the one from the law of cosines with different parameter mappings than for D_2 : $len1$ as a , $len2$ as b , and $dist$ as c .



And that's it. Let's pour this into code now.

3.1 The code

```
import (
    "fmt"
    "math"
)
const (
    len1 = 10.0
    len2 = 10.0
)
func lawOfCosines(a, b, c float64) (C float64) {
    return math.Acos((a*a + b*b - c*c) / (2 * a * b))
}
func distance(x, y float64) float64 {
    return math.Sqrt(x*x + y*y)
}
func angles(x, y float64) (A1, A2 float64) {
    dist := distance(x, y)
    D1 := math.Atan2(y, x)
    D2 := lawOfCosines(dist, len1, len2)
    A1 = D1 + D2
    A2 = lawOfCosines(len1, len2, dist)

    return A1, A2
}
func deg(rad float64) float64 {
    return rad * 180 / math.Pi
}
func main() {

    fmt.Println("Lets do some tests. First move to (5,5):")
    x, y := 5.0, 5.0
    a1, a2 := angles(x, y)
    fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))

    fmt.Println("If y is 0 and x = Sqrt(10^2 + 10^2), then alpha should become 45 degrees and beta should become 90 degrees.")
    x, y = math.Sqrt(200), 0
    a1, a2 = angles(x, y)
    fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))

    fmt.Println("Now let's try moving to (1, 19).")
    x, y = 1, 19
    a1, a2 = angles(x, y)
    fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))

    fmt.Println("n extreme case: (20,0). The arm needs to stretch along the y axis.")
    x, y = 20, 0
    a1, a2 = angles(x, y)
    fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))
}
```



```

fmt.Println("And (0,20).")
x, y = 0, 20
a1, a2 = angles(x, y)
fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))

```

```

fmt.Println("Moving to (0,0) technically works if the arm segments have the same length, and if the arm does not block itself. Still the result looks a bit weird!?)")

```

```

x, y = 0, 0
a1, a2 = angles(x, y)
fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))

```

```

fmt.Println("What happens if the target point is outside the reach? Like (20,20).")

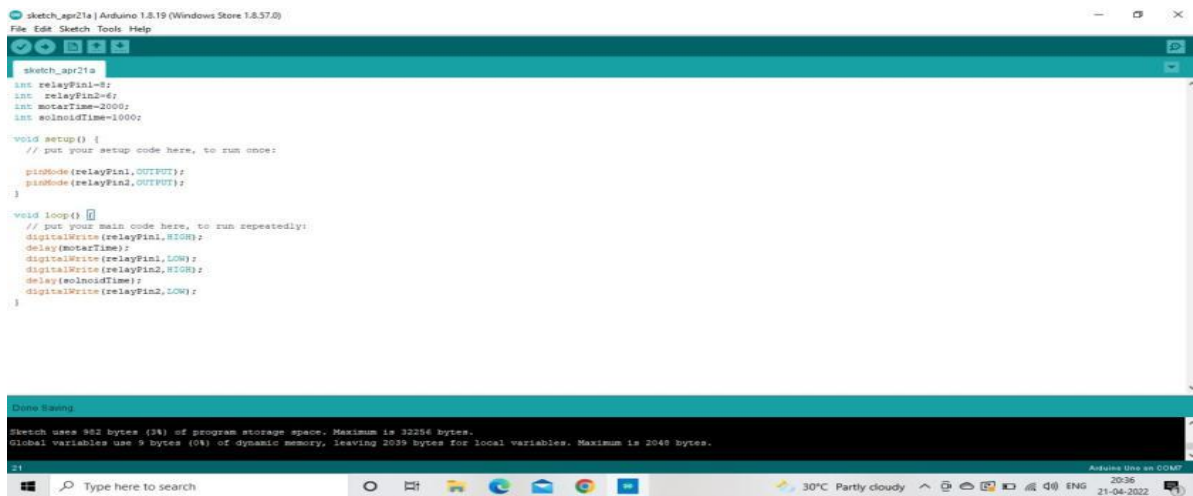
```

```

x, y = 20, 20
a1, a2 = angles(x, y)
fmt.Printf("x=%v, y=%v: A1=%v (%v°), A2=%v (%v°)\n", x, y, a1, deg(a1), a2, deg(a2))
}

```

This is the code used to run the robotic arm.



```

sketch_apr21a | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help

sketch_apr21a
int relayPin1=5;
int relayPin2=6;
int motorTime=2000;
int solenoidTime=1000;

void setup() {
  // put your setup code here, to run once:
  pinMode(relayPin1,OUTPUT);
  pinMode(relayPin2,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(relayPin1,HIGH);
  delay(motorTime);
  digitalWrite(relayPin1,LOW);
  digitalWrite(relayPin2,HIGH);
  delay(solenoidTime);
  digitalWrite(relayPin2,LOW);
}

```

Done Saving

Sketch uses 462 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2029 bytes for local variables. Maximum is 2048 bytes.

4. CALCULATIONS

Arm Lengths

Between each motor joint is an arm linkage L. Enter the length of each linkage. If a linkage does not exist in your design, set L to zero.

Select inches or meters	inches
L1	5
L2	5
L3	5

Arm Weight

Now enter the weight of each arm linkage. The center of mass is assumed to be 1/2 of L. If a linkage does not exist, set the weight to zero.

Select pounds or kilograms	pounds

W2	<input type="text" value="1"/>	W4	<input type="text" value="1"/>
W6	<input type="text" value="1"/>	W7 (object weight)	<input type="text" value="1"/>

Motor Weight

Enter in the weight of each motor. Motor 1 is the base motor, Motor 2 is the middle joint, and Motor 3 is the wrist motor.

Select pounds or kilograms	<input type="text" value="pounds"/>
Base Motor M1	<input type="text" value="1"/>
Joint Motor M2	<input type="text" value="1"/>
Wrist Motor M3	<input type="text" value="1"/>

Joint Rotational Acceleration

Note: For some reason the result when adding in acceleration looks astronomically high, but I can't figure out for the life of me where my equation mistake is. Just leave these at 0 if you don't trust the result. Sorry! Look at my source if you think you can figure it out . . . For each joint to rotate at a specific acceleration, you need to add additional torque to what you need just for static lifting. Fill in the acceleration you want for each joint.

V0	<input type="text" value="30"/>	degrees/s^2
V1	<input type="text" value="30"/>	degrees/s^2
V2	<input type="text" value="30"/>	degrees/s^2
V3	<input type="text" value="30"/>	degrees/s^2

Motor Efficiencies

Motors and gearing are never 100% efficient. Enter expected efficiency. If you are unsure about efficiency, check out my gearing efficiency tutorial.

M1	<input type="text" value="90"/>	%	M2	<input type="text" value="90"/>	%	M3	<input type="text" value="90"/>	%
-----------	---------------------------------	----------	-----------	---------------------------------	----------	-----------	---------------------------------	----------

Torque Result

These are the finished results. This is the maximum torque that each motor requires to lift both the arm and the given object weight at full extension at required velocity. Shorter arms and lower weights reduce required torque.

Select Units	<input type="text" value="pound*inch"/>
---------------------	---

Motor 0 Torque M0	<input type="text"/>
Motor 1 Torque M1	<input type="text"/>
Motor 2 Torque M2	<input type="text"/>
Motor 3 Torque M3	<input type="text"/>

Misc Results

Useful information to help you with other parts of your robot design. For the encoder calculation, this will tell you the expected typical positioning error due to the base motor encoder. If you are using a servo for the base motor, enter 360. Shorter arms and higher resolutions decrease error.

Total Weight	<input type="text"/>	<input type="text" value="lb"/>
Enter Base Motor Encoder Res.	<input type="text" value="360"/>	Ouput Arm Accuracy (inches) <input type="text"/>

5. FUTURE SCOPE OF ROBOTIC ARM

- The world has entered the era of Industry 4.0. robots assist human manufacturing, emphasizing the use of "human-machine collaboration" to move toward smart production Enterprises have deployed automation equipment to improve production efficiency
- In the medical field major operations are being done by the robotic arm's already.
- Robotics and Machine Vision are different in professional research fields. Robotics usually belong to the fields of mechanical engineering and automatic control engineering; while machine vision belongs to information engineering and electrical engineering. Through the cooperation of experts in these two major fields, the robot can be given the ability of visual perception.

It can be seen that robot vision is a highly integrated engineering technology that detects people and objects in the environment through machine vision, calculates their position on the camera coordinate system, converts to the robotic arm coordinate system, and then drives the motor Driving the shaft joint to operate the target is a seemingly simple process, but in fact, it implies complex computer operations.

6. CONCLUSIONS

By all the information with the calculations and the practical considerations we conclude that the future of the robots will bring drastic changes in the production and medical fields .the human effort will be decreased and the robotic engagement on work will be increased without destroying the labour market .And also Robots facilitate disaster response, augment physical abilities, serve in areas where there's a need for interaction with people, and enable exploration beyond the boundaries of Earth. Robotics has applications not only in the field of manufacturing or assembly lines.

REFERENCE

- [1] WMHW Kadir, RE Samin, BSK Ibrahim. Internet controlled a robotic arm. Procedia Engineering. 2012.
- [2] MAK Yusoff, RE Samin, mobile robotic arm. Procedia Engineering. 2012.
- [3] AM Al-Busaidi, Development of an educational environment for online control of a biped robot using MATLAB