Mechatronics Engineering
(MTE) Program

Faculty of Engineering
Mansoura University

Project 2

Multi-tasking SCARA Robot

Team Members:
Abdelrahman Ahmed Mostafa
Ahmed Mohamed Eissa
Bahy Hatem Hassan Abolfotoh
Hany Samir Abdellatif Atia
Hesham Ahmed Hassan
Khaled Tarek Fathy
Mahmoud Essam Mohamed
Mohamed Alaaeldin Gaafar
Mohamed Elhadidy
Mohamed Elnady
Mohamed Hataba
Mohamed Mohsen Salama
Mohamed Tarek Fawzy
Mustafa Elsherbiny

Supervision of:

Dr. Fatma Elerian
Eng. Mostafa Awad Allah

2023

Abstract

The robot has 4 degrees of freedom and it’s driven by 4 NEMA 17 stepper motors. Additionally, it has a small servo motor for controlling the end effector or the robot gripper in this case. The brain of this SCARA robot is an Arduino UNO board which is paired with a CNC shield and four A4988 stepper drivers for controlling the stepper motors.

Using the Processing development environment, we made a Graphic User Interface which features both Forward and Inverse Kinematics control. With the Forward Kinematics we can manually move each robot joint in order to get the desired position. Using the sliders on the left side, we can set the angle of each joint. The final position of the end effector, the X, Y and Z values are calculated and printed on the right side of the screen.

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have these all along our way to complete our project. All that we have done is only due to such supervision and assistance, and we won’t forget to thank them.

We would like to express our thanks and deepest appreciation to our project supervisor Dr. Fatma Elerian who didn’t keep any effort in encouraging us to do such a great job.

Last but not least, we are very thankful to our der parents for their continuous support at all the time in order to see this moment of our first project on the university, we always wish you happiness and loge life.

Table of Contents

List of Figures

List of Tables

Phase 1: Project Definition

1.1 Introduction

The SCARA is a type of industrial robot. The acronym stands for Selective Compliance Assembly Robot Arm[1] or Selective Compliance Articulated Robot Arm.[2]

By virtue of the SCARA's parallel-axis joint layout, the arm is slightly compliant in the X-Y direction but rigid in the Z direction, hence the term selective compliance. This is advantageous for many types of assembly operations, for example, inserting a round pin in a round hole without binding.

The second attribute of the SCARA is the jointed two-link arm layout similar to human arms, hence the often-used term, articulated. This feature allows the arm to extend into confined areas and then retract or "fold up" out of the way. This is advantageous for transferring parts from one cell to another or for loading or unloading process stations that are enclosed.

SCARAs are generally faster than comparable Cartesian robot systems. Their single pedestal mount requires a small footprint and provides an easy, unhindered form of mounting. On the other hand, SCARAs can be more expensive than comparable Cartesian systems and the controlling software requires inverse kinematics for linear interpolated moves. However, this software typically comes with the SCARA and is usually transparent to the end-user.

Sankyo Seiki, Pentel and NEC presented the SCARA robot as a completely new concept for assembly robots in 1981. The robot was developed under the guidance of Hiroshi Makino,[3] a professor at the University of Yamanashi.[2] Its arm was rigid in the Z-axis and pliable in the XY-axes, which allowed it to adapt to holes in the XY-axes.

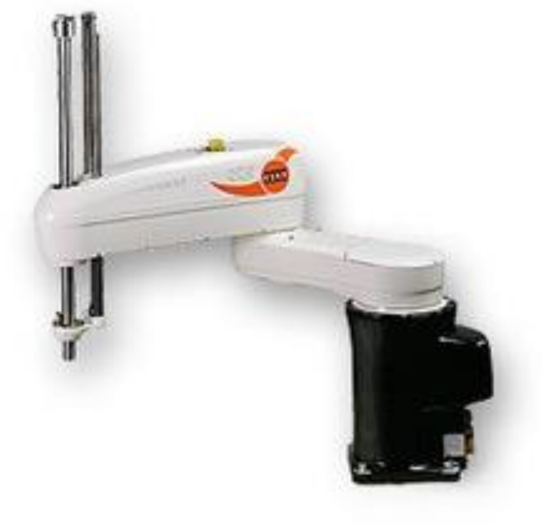


FIGURE 1

Typical SCARA robot, made by KUKA

1.2 Industrial Robots [4]

1.2.1 Robot Structures

An industrial robot is typically some form of jointed structure of which there are various different configurations. The robot industry has defined classifications for the most common and these are:

- Articulated
- SCARA
- Cartesian
- Parallel (or Delta)
- Cylindrical.

These structures and their benefits are described in more detail below. The structures are achieved by the linking of a number of rotary and/or linear motions or joints. Each of the joints provides motion that collectively can position the robot structure, or robot arm, in a specific position. To provide the ability to position a tool, mounted on the end of the robot, at any place at any angle requires six joints, or six degrees of freedom, commonly known as six axes.

The working envelope is the volume the robot operates within. This is typically shown (see Figure 2.1) as the volume accessible by the centre of the fifth axis. Therefore, anywhere within this working envelope the robot can position a tool at any angle. The working envelope is defined by the structure of the robot arm, the lengths of each element of the arm, and the motion type and range that can be achieved by each joint. The envelope is normally shown as a side view, providing a cross-section of the envelope, produced by the motion of axes 2–6 and a plan view then illustrating how this cross-section develops when the base axis, axis 1, is moved. It should also be noted that the mounting of any tools on the robot will also have an impact on the actual envelope accessible by the robot and tool combined.

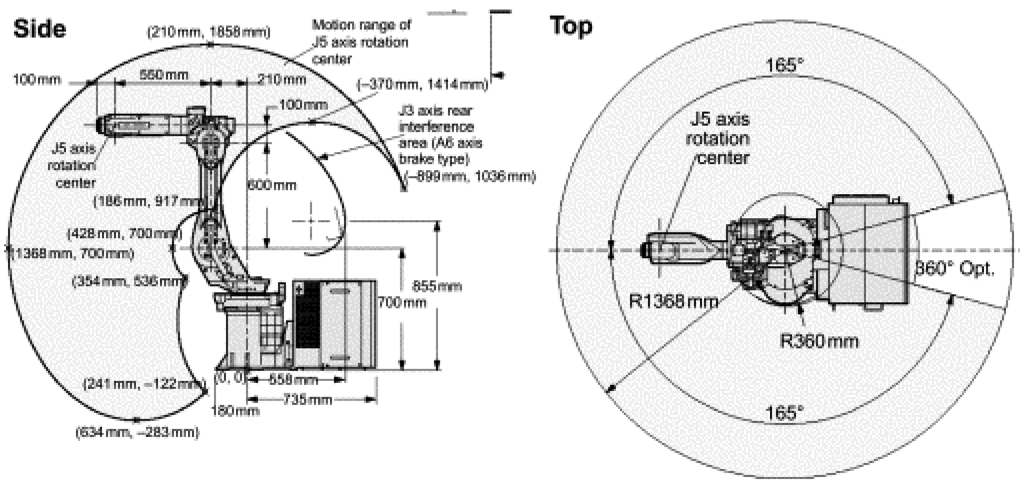


FIGURE 2

Typical working envelope.

The first robot, a Unimate, was designated as a polar-type machine. This design was particularly suited to the hydraulic drive used to power the robot. The robot (Figure 2.2) provided five axes of motion; that is, five joints that could be moved to position the tool carried by the robot in a particular position. These consisted of a base rotation, a rotation at the shoulder, a movement in and out via the arm, and two rotations at the wrist. The provision of only five axes provided limitations in terms of the robot's ability to orientate the tool. However, in the early days, the control technology was unable to meet the needs for six axes machines.

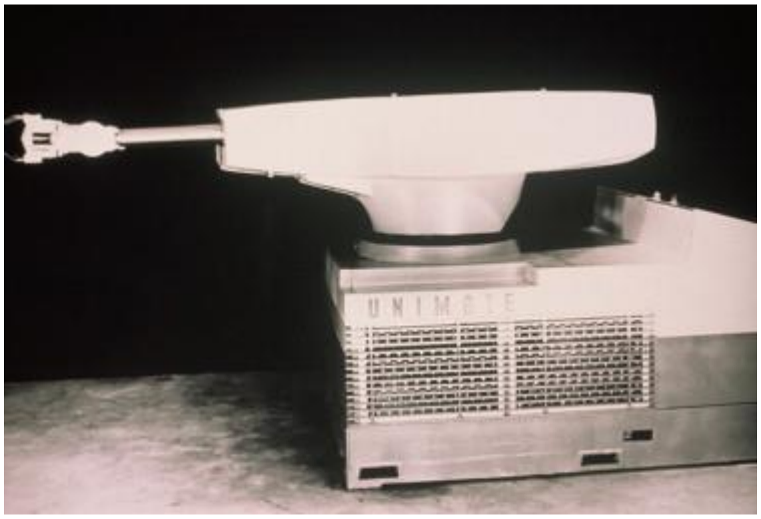


FIGURE 3

Unimate robot.

1.2.1.1 Articulated Arm

The most common configuration is the articulated or jointed arm (Figure 2.3). This closely resembles the human arm and is very flexible. These are normally six axis machines, although some seven axis machines are available, providing redundancy and therefore improving access into awkward spaces. The structure comprises six rotational joints, each mounted on the previous joint. They have the ability to reach a point, within the working envelope, in more than one configuration or position a tool in any orientation at a specific position.

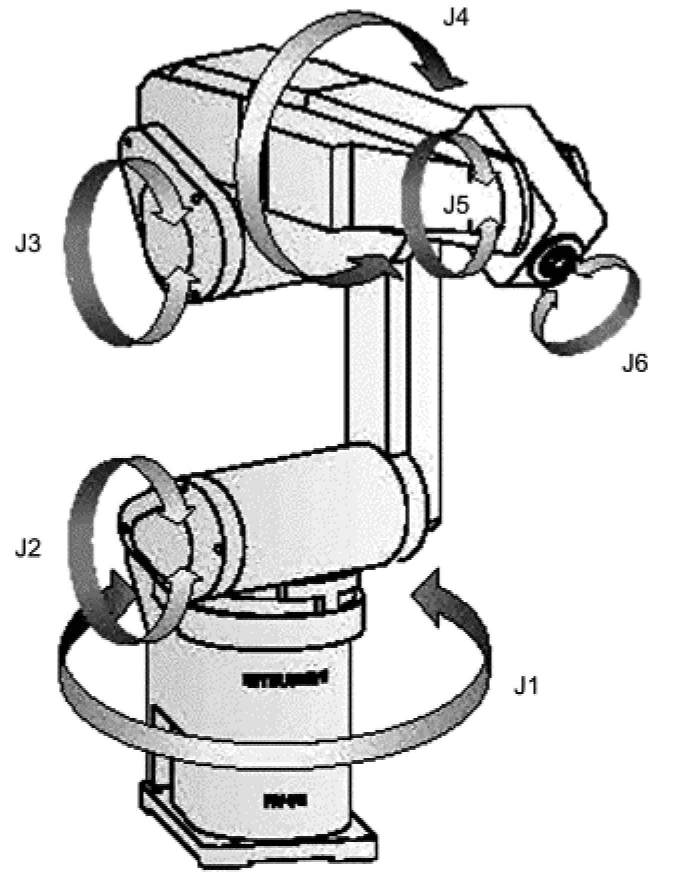


FIGURE 4

Jointed arm configuration.

The joint motion of articulated arm robots is complex and therefore can be difficult to visualise. The construction of the arm means each joint has to carry the weight of all the following joints; that is, joint three carries 4, 5, and 6. This impacts both the carrying capacity, the load that can be handled by the robot, as well as the repeatability and accuracy (see Section 2.2). The structures are not particularly rigid and the overall repeatability is the cumulative of all of the axes. However, the increasing performance of AC servo motors and the improvement in the mechanics provide excellent performance for the majority of applications.

As mentioned, the articulated arm is the most common industrial robot structure providing about 60% of annual installations worldwide, although it is higher in Europe and the Americas (International Federation of Robotics, 2013). This type of robot is used for many process applications, including welding and painting, as well as many handling applications including machine tool tending, metal casting, and general material handling. Typical robot sizes range from a reach of 0.5 to over 3.5 m and carrying capacities from 3 to over 1000 kg.

There are also a number of four axis articulated arms. These have been developed specifically for applications such as palletising, packing, and picking where it is not necessary to orientate the tool. Therefore two of the wrist axes are not required. This type of robot is able to achieve higher speeds with higher payloads than the equivalent six axes machines.

Dual arm robots, with two articulated arms mounted on the same structure, are also being developed. These two arms are able to work cooperatively and therefore mimic a human and are aimed at tasks such as assembly where two hands are required to work together to assemble the parts.

1.2.1.2 SCARA

The SCARA configuration (Figure 2.4) provides different attributes to the articulated arm. This configuration was originally developed for assembly applications, hence the name Selective Compliance Assembly Robot Arm. The four-axis arm includes a base rotation, a linear vertical motion followed by two rotary motions in the same vertical plane. Due to the nature of the configuration the arm is very rigid in the vertical direction and can also provide compliance in the horizontal plane. It provides high speed combined with high acceleration and works to very tight tolerances.

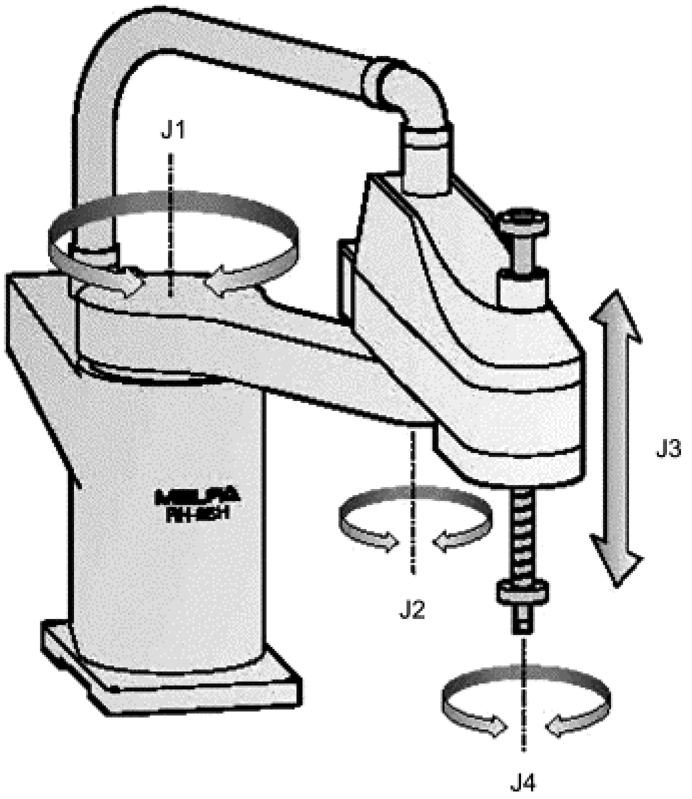


FIGURE 5

SCARA configuration.

SCARA machines are typically small with the largest having carrying capacities of about 2 kg and a reach of about 1 m. They are mainly used for assembly applications although they can also be used for packing, small press tending, adhesive dispensing, and other applications. The application of SCARAs is mainly constrained by their size and the limitation of being only four axes.

A standard cycle time test for robots has been defined to provide comparability between machines. This so-called goalpost test consists of a 25 mm vertical move upward, followed by a 300 mm horizontal move, and a 25 mm vertical move down and simulates a typical move for an assembly application. The timings for SCARA robots to achieve this move, both forward and return, can be as low as 0.3 s. This is normally faster than the equivalent articulated arm, six-axis robots.

The SCARA configuration makes up about 12% of global sales although it is more popular in Asia, due to the size of the electronics sector in this region. Asia accounts for about 50% of all SCARA robot sales (International Federation of Robotics, 2013).

1.2.1.3 Cartesian

The cartesian category encompasses all industrial robots that include only linear drives for their three major axes (Figure 2.5) and the motions are coincident with a cartesian coordinate system. These machines are often limited to three axes, although some special versions have been developed with additional rotary axes mounted on the last linear axis. This cartesian category includes gantry machines as well as linear pick and place devices. The configuration of these are varied and they can also be constructed from modular kits, providing the flexibility to design a machine for a specific requirement. Gantries can be goalpost type devices, supported on one structure only, as well as area gantries with two support structures. The main axis can range in length from less than 1 m to many tens of metres. Gantries can also be very heavy duty, able to carry 3000 kg. A further benefit of gantries is that they minimise the impact on the factory floor and manual access to machines, as they are largely overhead. However, they are often more expensive than the equivalent articulated arm robots.

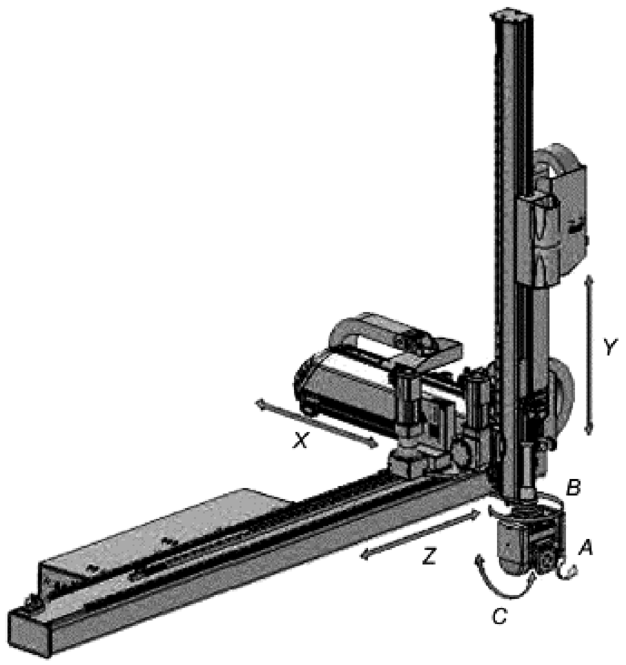


FIGURE 6

Cartesian configuration.

Applications are quite varied although they are typically used for handling, palletising, plastic moulding, assembly and machine tending. They also have some application for processes such as welding and glueing, particularly on very large parts. Cartesian machines are the second most popular configuration, taking about 22% of global robot sales (International Federation of Robotics, 2013).

1.2.1.4 Parallel

The parallel or delta robot configuration (Figure 2.6) is one of the most recent configuration developments. This includes machines whose arms have concurrent prismatic or rotary joints. These were developed as overhead mounted machines with the motors contained in the base structure driving linked arms below. The benefit of this approach is that it reduces the weight within the arms and therefore provides very high acceleration and speed capability. However they do have a low payload capacity, typically under 8 kg.

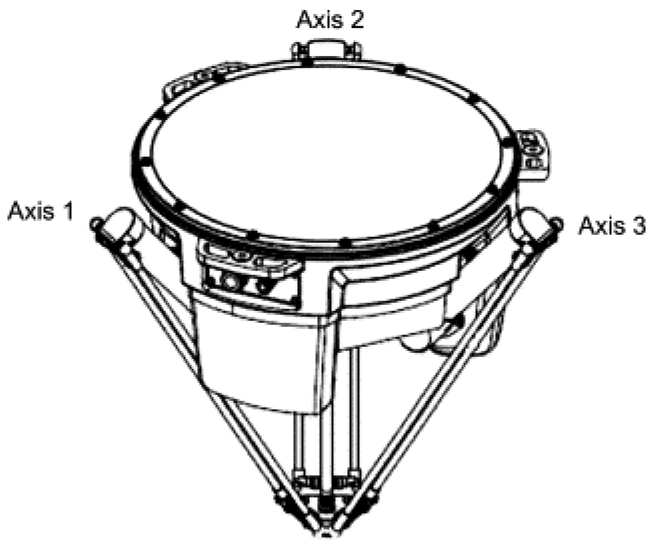


FIGURE 7

Parallel configuration.

Therefore, the main application is picking, particularly on packing lines for the food industry, and also assembly applications. These machines can achieve similar cycle times to the SCARAs with the fastest achieving the goalpost test (25, 300, 25 mm) in 0.3 s. This type of robot is sold in relatively small numbers, achieving only about 1% of the global market (International Federation of Robotics, 2013).

1.2.1.5 Cylindrical

These robots have a combination of rotary and linear axes, typically with a base rotation followed by a vertical and horizontal linear axis and further rotary axes at the wrist. They provide a rigid structure, with good access into cavities and are easy to programme and visualise. However, they do require clearance at the rear of the arm. They are particularly suited to machine tending and general pick and place applications.

They are mainly used in the electronics industry, particularly clean room applications, and take about 2% of the global market. Similar to the SCARA, they are most popular in Asia, due to the strength of the electronics sector in that region, which takes about 90% of global sales (International Federation of Robotics, 2013).

1.3. Needs/Problems

Every project exists to fill a need or solve a problem, project needs/problems are the driving force of the project. Following an analysis of needs and problems.

1.3.1. Industrial Robots Market Size and Share

The global industrial robots market size was valued at USD 15.60 billion in 2021. The market is projected to grow from USD 16.78 billion in 2022 to USD 35.68 billion by 2029, exhibiting a CAGR of 11.4% during the forecast period. The global COVID-19 pandemic has been unprecedented and staggering, with experiencing lower-than-anticipated demand across all regions compared to pre-pandemic levels. Based on our analysis, the global market exhibited a growth of 5.9% in 2020 as compared to 2019.

An industrial robot is a sort of mechanical equipment that is fed data and is programmed to execute activities linked to industry production automatically. These robots can be programmed, and the program may be modified as many times as required, depending on the application. This type of robots aid in improving productivity while lowering costs and generating high-quality goods in automation applications. Drives, end-effectors, robotic manipulators, sensors, and controls make up the majority of robots. The robotic controller is the robot's brain that aids in the delivery of commands. Microphones and cameras are used as robot sensors to keep the robot aware of the industrial surroundings.

The robotic manipulator is the robot's arm that assists the robot in moving and positioning, while the end effectors assist the robot in engaging with the workpieces. Cartesian, Collaborative, SCARA, Articulated, Cylindrical robots are types of robots utilized in the industry. The degree of freedom of movement, size requirements, and payload capacity all influence the type of robot chosen. Industrial robots aid in the optimization of production processes for a healthy and efficient outcome.

1.3.2. COVID-19 IMPACT

Positive Surge in Demand for Industrial Robots During COVID-19

The continuing shift from manual to automated operations is driving up demand for these robots. The synchronization of management, production, and control is critical to the seamless operation of an industry's workflow. During the manufacturing of robots, however, interruption and inefficiency in any of these roles might impede operations and product quality. In such a situation, robotic technology is gaining traction since it streamlines procedures and improves workflow efficiency. Due to the growth of small and medium-sized businesses, increased expenditures on automation across sectors, and strict regulatory rules governing the handling of dangerous chemicals and goods, the need for robots is speeding up.

Similarly, these robots aid in high-payload lifting during vehicle production and machinery customization. The rise of smart factories will expand market opportunities. Furthermore, factors such as increased public awareness of industrial accidents, expanding consumer goods demand, and employee safety are all contributing to the industrial robots market growth. Manufacturers are concentrating their efforts on research & development to incorporate artificial intelligence and build advanced sensors, which will boost the market growth. Venture investors are taking notice of market changes and are eager to invest in companies that develop, test, and produce robots.

Furthermore, manufacturers are pursuing different business strategies such as mergers, acquisitions, and cooperation to broaden their global reach and meet market capitalization. For example, Teradyne, Inc. purchased Mobile Industrial Robots (MiR) for USD 147 million in April 2018. Teradyne's existing range of industrial type of robots and sophisticated intelligent automation technologies will be expanded as a result of this strategic decision.

1.3.3. LATEST TRENDS

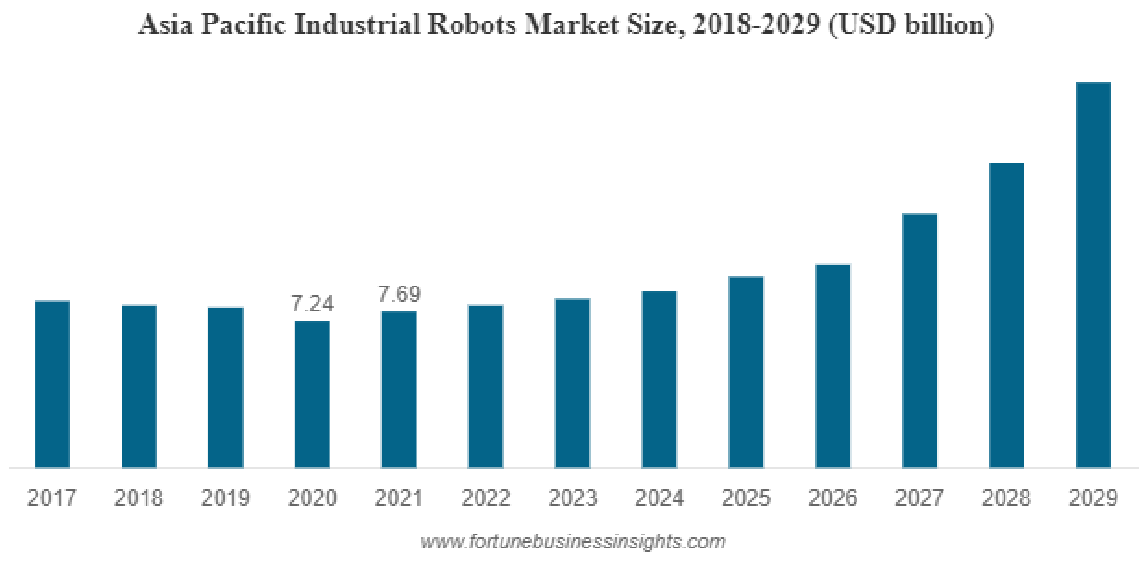


FIGURE 8

LATEST TRENDS

Rise in the Logistics Sector is Expected to Aid the Market Consumers' increasing preference for purchasing online is helping the e-commerce business. The integration of automated robotic systems is becoming a priority for distribution centers, merchants, warehouses, and facility owners. The goal is to deliver on time, save money on labor, and improve efficiency and production in every step. This is one of the most important considerations for the deployment of these robots. Many firms are also creating interesting solutions throughout the whole logistics value chain, such as selfdriving trucks, intelligent warehouses, and service robots.

1.3.4. DRIVING FACTORS

In general, industrial robots are in high demand in industries such as automotive, pharmaceuticals, consumer electronics, packaging, and equipment. This need, however, is predicated on the sort of robot that they need to place throughout their locations to harness industrial activity and cut costs. Players in the consumer electronics industry, for example, could deploy collaborative robots at a faster rate to increase production flexibility.

As a result, there are significant investments being made in the business, which is increasing the need for these robots. For example, Nissan Motor purchased two lines of UR10 collaborative robots i.e. arms for its Yokohama facility from Universal Robots in order to cut labor costs while maintaining the efficiency of manufacturing procedures such as Takt Time. Similarly, pharmaceuticals, consumer electronics, & industrial companies all have a stake in the market which has contributed to the market growth.

1.3.5. RESTRAINING FACTORS

High Costs of Investments at Initial Stage and Maintenance Hinder the Market Growt

For organizations with little or no relevant expertise, the initial investment necessary might be difficult. Procurement, integration, programming, accessories, and maintenance, among other things, necessitate a large capital investment. This might stifle market expansion. Due to low-volume manufacturing and a poor return on investment, small and medium businesses also find it difficult to park large sums of money (ROI).

Moreover, the recent collapse of the COVID-19 pandemic is having a significant influence on the market. The quick shutdown of different industrial units and businesses has impacted overall production and supply networks significantly. Industrial robot makers, on the other hand, are projected to see a boost in sales and overcome the obstacles following the pandemic.

1.3.7. SEGMENTATION

By Robot Type Analysis

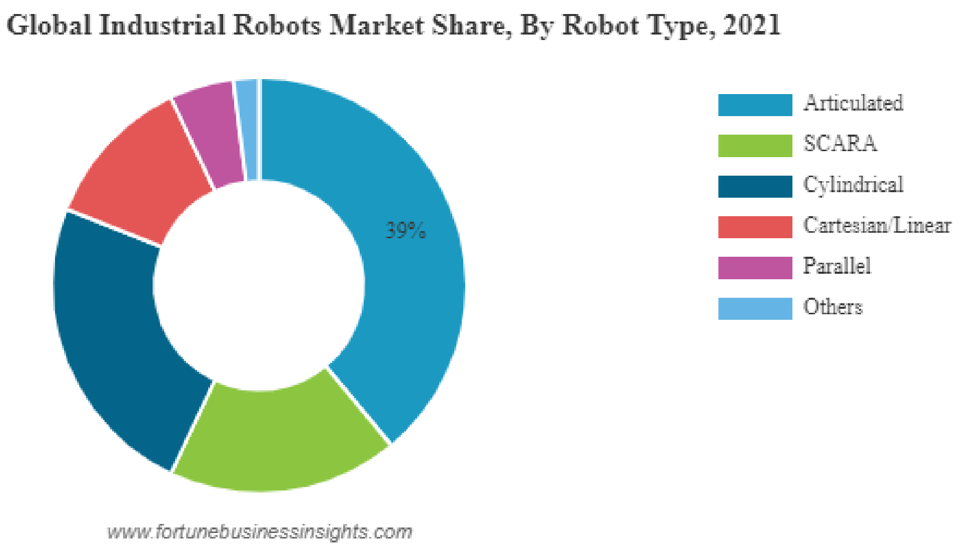


FIGURE 9

Global Industrial Robots Market Share

Articulated Robots Segment is Predicted to Show a Higher Growth Rate Due to Growing Penetration

Based on robot type, the market is divided into SCARA, articulated, cartesian/linear, cylindrical, parallel, and others.

Articulated robot has the largest industrial robots market share due to its ability to conduct exact motions repeatedly, which is used in applications such as material handling, assembly, and others. The employment of articulated robots in food and beverage, pharmaceuticals, and autos is propelling the market's growth.

Due to its construction built to accomplish duties in a hostile environment, SCARA and cylindrical robots are predicted to expand significantly throughout the projection period. Furthermore, they conduct a variety of selective and repetitive tasks, such as pick and place, sorting, prescription medicine dispensing, and tiny part assembly, all with a short cycle time. Robots of this type are frequently used in the food and beverage, automotive, and industrial industries.

Cartesian/linear and parallel robots are expected to see significant expansion in the future years as warehouses, distribution centers, and manufacturing facilities become more automated.

By Application Analysis

Material Handling Segment to Grow Rapidly during the Forecast Period Based on application, the market is classified into welding & soldering, pick & place, assembling, material handling, cutting & processing, and others Material handling is likely to see progress during the forecast period due to demand from the food and beverage, pharmaceutical, and electrical and electronics industries. Furthermore, as certain chemicals are concentrated and transport hazardous compounds, material handling is a key role in the chemical and pharmaceutical industries. In such a situation, robots are frequently used to decrease accidents and maintain staff safety.

During the forecast period, assembly, welding & soldering and pick and place applications are likely to have a significant increase. A surge in the automotive, metal industrial, electrical, and electronic industries, where bespoke spare parts are built and placed in the completed product, is expected to boost growth.

With the rising trend of automotive personalization and the increased focus on precision cutting across many applications, the cutting and processing industry is likely to rise significantly in the future years.

By Industry Analysis

Healthcare & Pharmaceutical to Develop Rapidly During Forecast Period Owing to Increasing Usage in Medical Operations Based on industry, the market is segregated into electrical & electronics, automotive, food & beverages, healthcare & pharmaceutical, metals & machinery, rubber & plastic, and others

The healthcare & pharmaceutical market is predicted to develop rapidly. The expanding trend of combining robots and humans to conduct diverse medical tasks is expected to fuel growth. This involves supporting surgeons with surgeries, which are usually less invasive. These robots use powerful 3DHD technology to provide the operator with the spatial references needed for very intricate operations. Furthermore, speed and precision are important attributes for rising industrial robot deployments. At testing labs and production companies, these robots can handle liquids, powder & even viscous materials.

Due to the huge growing move from manual to electric vehicles, the automotive industry is predicted to develop quickly. Furthermore, regulatory authorities' growing worries about reducing emissions compel manufacturers to lower vehicle weight in order to meet emission regulations. In addition, major countries such as Germany, the U.K., the U.S., and China are focusing on extending their manufacturing units across borders to fulfill increased demand.

With increasing integration of robotics for handling delicate semiconductors, uncertain chemicals in research & development centers, and other applications, the metals & machinery, electrical & electronics, and food & beverage segments are expected to grow significantly in the future. In addition, with the increased need for specialized and modified machinery, industrialization has picked up speed in recent years. To withstand the enormous strain in the operational operations, these machines require precise soldering & welding.

1.3.7. REGIONAL INSIGHTS

Asia Pacific Industrial Robots Market Size, 2021 (USD billion)

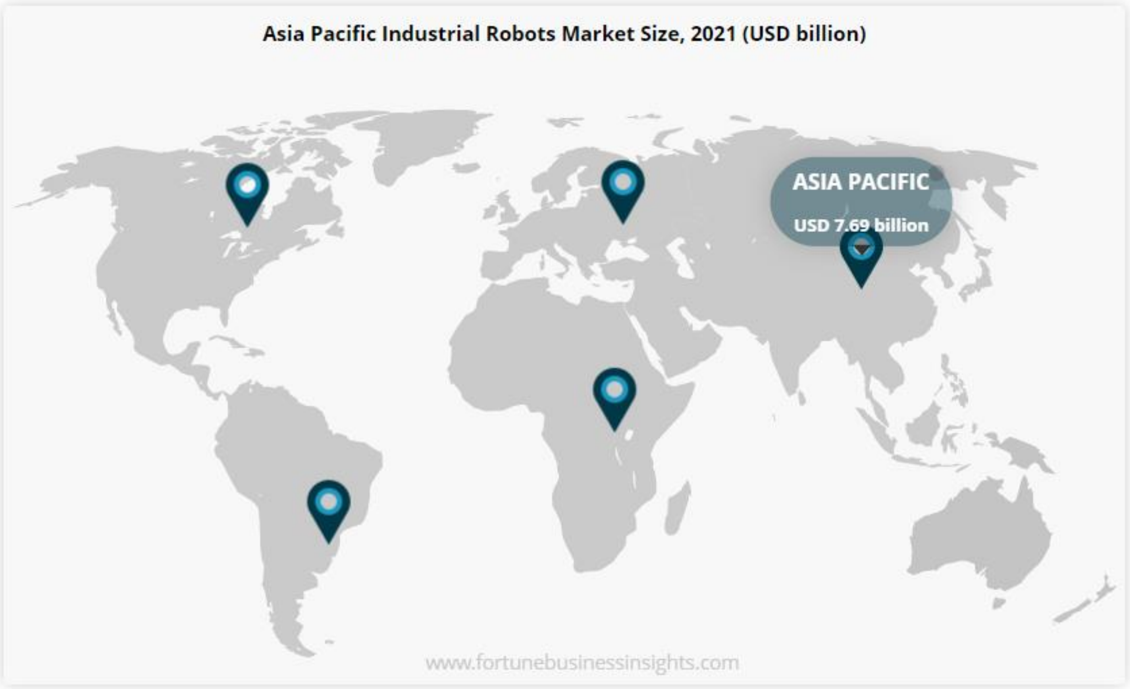

Asia Pacific Industrial Robots Market Size, 2021 (USD billion)
ASIA PACIFIC
USD 7.69 billion
www.fortunebusinessinsights.com

FIGURE 10

REGIONAL INSIGHTS

The report's scope comprises five major regions, North America, Europe, Asia Pacific, the Middle East & Africa, and South America Asia Pacific is expected to dominate the worldwide market during the forecast period. Rising automation, particularly in Japan, China, and India, is expected to fuel growth in the area. Furthermore, as the population and disposable income of people rise, so does the demand for consumer goods and other items, resulting in increased production capacity, which adds to the growth of this industry. Similarly, there is a growing need for bespoke machinery and autos, which necessitates high accuracy and rapid manufacturing capacity. The use of robots in industrial applications is increasing in order to keep up with demand, further increasing the global market share.

In terms of both value and volume, China is predicted to have the greatest share of the market for these robots in 2021. This is due to rising labor costs in APAC, which are prompting firms to automate their production processes in order to preserve their cost advantage. By 2021, China is likely to recover from its industrial slump, while Taiwan and Thailand strive for even more automation. Due to factors, such as cheap manufacturing costs, easy availability of inexpensive labor, lax pollution and safety standards, and government's push for foreign direct investments (FDIs), APAC is positioned to retain a strong market position in the conventional robotics industry in the foreseeable future.

North America is predicted to rise significantly as the worldwide market is driven by the advent of smart factories and industry 4.0, resulting in increased demand for customized and small robotic systems. Furthermore, as the manufacturing process is hastened and production capacity is increased, producers get an early return on investment (ROI).

Manufacturers are working on creating and adopting collaborative robots in Europe, which is likely to exhibit significant growth in the future years as the up-gradation of industrial operations is at an all-time high in this area. These robots are particularly built to operate with people and are equipped with advanced sensors that allow them to recognize and interact with them. This sort of robot is employed to do heavy-duty tasks.

The Middle East & Africa and Latin America are expected to show substantial growth in the near future. The growth is attributed to the supply of the bulk of these robots from the automotive industry. This has prompted businesses in the electronics, food and beverage, and aerospace industries to use robots for a variety of tasks. In recent years, corporations, such as Philips, Ford, GM, and Nestle, have introduced robotic work cells. In April 2021, Accenture purchased Pollux, a Brazilian firm that specializes in industrial robots and automation solutions, to strengthen its digital manufacturing, operations, and supply chain capabilities. Furthermore, the increased utilization of the robots is fueling the growth of this market in Brazil.

1.3.8. KEY INDUSTRY PLAYERS

Key Players are Focusing on Upgrading the Existing Technology and Geographical Expansion Strategies

Key suppliers are concentrating on introducing its product streamlined with innovative technologies to gain recognition in the global market. For instance, in February 2021, ABB cobot families to offer higher payloads and speeds. Also, owing to the growing need for robotics technology, the players are emphasizing reaching out to the various industries across the potential geographies to uplift their customer base and presence. Apparently, Fanuc Corporation invested USD 240 million to expand its business presence in Shanghai (China) with the expectation to maintain its high edge competition in the China market.

LIST OF KEY COMPANIES PROFILED:

- ABB (Switzerland)
- YASKAWA ELECTRIC CORPORATION (Japan)
- Mitsubishi Electric Corporation (Japan)
- NACHI-FUJIKOSHI CORP. (Japan)
- Comau SpA (Italy)
- KUKA AG (Germany)
- FANUC CORPORATION (Japan)
- DENSO CORPORATION (Japan)
- Kawasaki Heavy Industries, Ltd. (Japan)
- Omron Corporation (Japan)

1.4. Scope

Project scope is the part of project planning that involves determining and documenting a list of specific project goals, deliverables, tasks, costs and deadlines. The following list is the scope statement of the project:

- Multi-tasking SCARA robot with gripper tool.

1.5. Goals and Objectives

✓ Build industrial multi-tasking SCARA robot.

1.6. Basic Economy

No.	Item	Price
1	Arduino Uno	300
2	Servo motor	300
3	4x NEMA 17 stepper motors	1300
4	CNC shield and 4x A4988 Stepper Driver	500
5	3D Printing	15000
6	Other (includes risks)	~4600
Total		~22000

TABLE 1

Basic Economy

Resources of component:
We can get approximately 5 resources as shown blow:

A. **3 Local resources:**
   1. Electronic Village
   2. Lampatronics
   3. UGE-one

B. **2 international resources:**
   1. Amazon

Resources of information and needed knowledge:
We have 3 main resources:

A. **Online resources:**

1. Elsevier
2. ResearchGate
3. The Egyptian Knowledge Bank

B. Companies may be interested at projects:

1. ABB. 2. KUKA

C. Our collage PHD and Professors:

professors of electrical and mechanical department.

Guestimate of needed time to collect all needed Info for project as shown blow

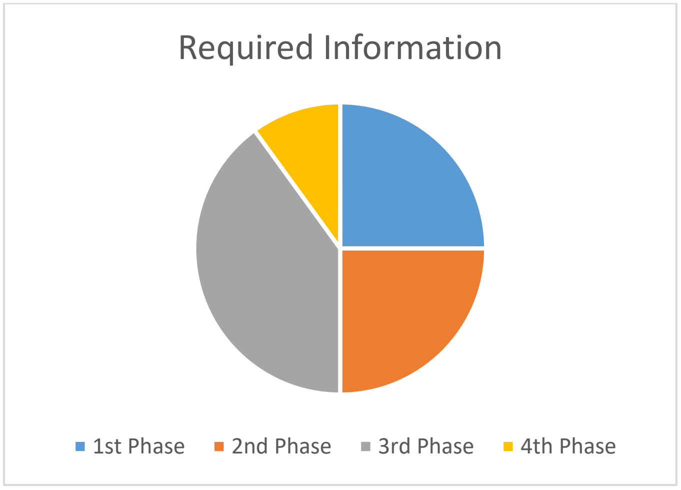


FIGURE 11

Required Information by project phases.

1.7. Stakeholders

“A person or group of people who have a vested interest in the success of an organization and the environment in which the organization operates”

- Team Members
- Supervisor of the project
- Professor and the head of the department of Mechatronics
- Dean of the College of Engineering
- President of Mansoura University
- Government
- Environment and nature
- Organisms and ecosystem
- Industries and companies
- Suppliers
- Energy and fuel company

Phase 2: Project Planning

2.1. Team Members

Team members are distributed on the following groups.

2.1.1. Project Manager

- Mohamed Alaaeldin Mohamed Gaafar

2.1.2. Mechanical Group

- Mohamed Alaaeldin Mohamed Gaafar
- Mustafa Elsaied Abdo Elsherbiny
- Bahy Hatem Hassan Abo El-fotouh
- Abdelrahman Ahmed Mostafa
- Hany Samir Abdellatif Atia
- Ahmed Mohamed Eissa

2.1.3 Hardware

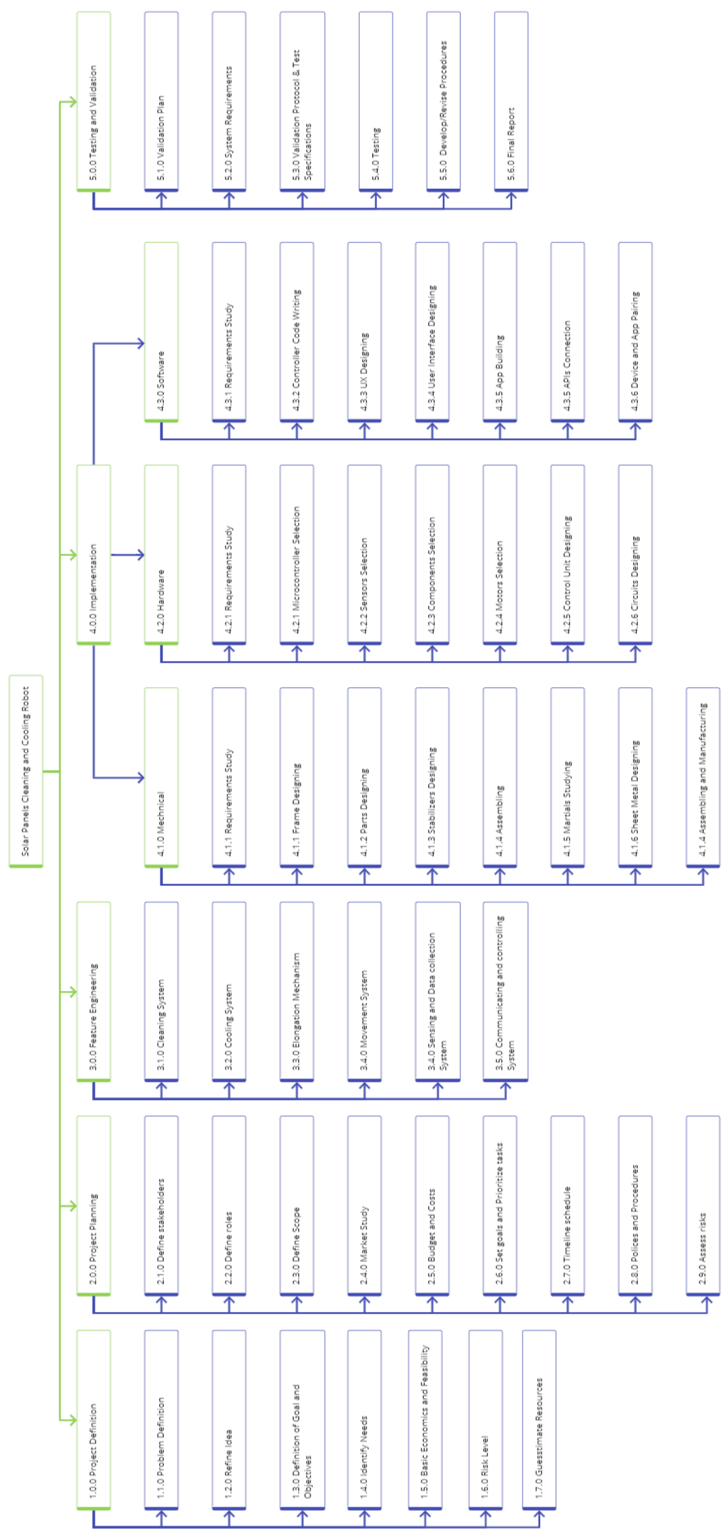
- Mohamed Alaaeldin Mohamed Gaafar
- Mohamed Mohamed Elnady
- Khaled Tarek Fathy Shaban
- Mohamed Mohsen Salama
- Hisham Ahmed Hasan

2.1.4. Software

- Mohamed Alaaeldin Mohamed Gaafar
- Mohamed Hataba
- Mohamed Yousry Othman Mohamed
- Mohamed Tarek Abdulmohsen
- Mahmoud Essam Mohamed

Thanks to our Dr. Fatma Elerian our team supervisor for her valuable advices and guidance.

2.2. Work Breakdown Structure



Solar Panels Cleaning and Cooling Robot
1.0.0 Project Definition
1.1.0 Problem Definition
1.2.0 Refine Idea
1.3.0 Definition of Goal and Objectives
1.4.0 Identify Needs
1.5.0 Basic Economics and Feasibility
1.6.0 Risk Level
1.7.0 Guesstimate Resources
2.0.0 Project Planning
2.1.0 Define stakeholders
2.2.0 Define roles
2.3.0 Define Scope
2.4.0 Market Study
2.5.0 Budget and Costs
2.6.0 Set goals and Prioritize tasks
2.7.0 Timeline schedule
2.8.0 Polices and Procedures
2.9.0 Assess risks
3.0.0 Feature Engineering
3.1.0 Cleaning System
3.2.0 Cooling System
3.3.0 Elongation Mechanism
3.4.0 Movement System
3.4.0 Sensing and Data collection System
3.5.0 Communicating and controlling System
4.0.0 Implementation
4.1.0 Mechanical
4.1.1 Requirements Study
4.1.1 Frame Designing
4.1.2 Parts Designing
4.1.3 Stabilizers Designing
4.1.4 Assembling
4.1.5 Martials Studying
4.1.6 Sheet Metal Designing
4.1.4 Assembling and Manufacturing
4.2.0 Hardware
4.2.1 Requirements Study
4.2.1 Microcontroller Selection
4.2.2 Sensors Selection
4.2.3 Components Selection
4.2.4 Motors Selection
4.2.5 Control Unit Designing
4.2.6 Circuits Designing
4.3.0 Software
4.3.1 Requirements Study
4.3.2 Controller Code Writing
4.3.3 UX Designing
4.3.4 User Interface Designing
4.3.5 App Building
4.3.5 APIs Connection
4.3.6 Device and App Pairing
5.0.0 Testing and Validation
5.1.0 Validation Plan
5.2.0 System Requirements
5.3.0 Validation Protocol & Test Specifications
5.4.0 Testing
5.5.0 Develop/Revise Procedures
5.6.0 Final Report

FIGURE 12

Project work breakdown structure

2.3. Risks

Risk Description	Likehood	Management action	Risk owner
Components unavailable in market	High	Prepare/Use local alternative	Market
Components damage	High	Secure spear components/fund	Team
Lack of fund	Low	Spend less	Team
Low quality components	High	Enhance the quality by software correction or relay on other components	Manufacturing companies/ Market
Design failure	Low	Test design before implementation, or change/fix design then and ensure safety	Team
Software Failure	Low	Software testing or fix software	Team
Lack of skilled craftsman	High	Create non-complex design	Market

TABLE 2

Project Risks

2.4. Time Plan

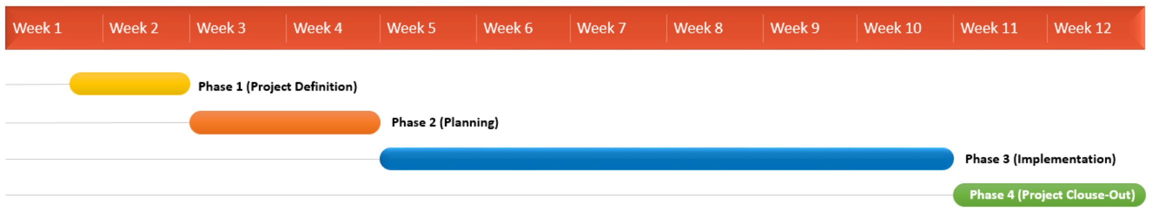


FIGURE 13

Time Plan

Phase 3: Implementation

CHAPTER 1: MECHANICAL

3.1. Mechanical Design

Mechanical design is to design parts, components, products, or systems of mechanical nature. For example, designs of various machine elements such as shafts, bearings, clutches, and gears fall into the scope of mechanical design.

3.1.1. *Tools and Requirements*

In order to design the mechanical system for the robot we used SOLIDWORKS software.

3.1.1.1 SOLIDWORKS

FIGURE 14

Solidworks logo

SolidWorks is a solid modeling computer-aided design (CAD) and computer-aided engineering (CAE) application published by Dassault Systèmes.

SolidWorks is a solid modeler, and utilizes a parametric feature-based approach which was initially developed by PTC (Creo/Pro-Engineer) to create models and assemblies. The software is written on Parasolid-kernel.

Parameters refer to constraints whose values determine the shape or geometry of the model or assembly. Parameters can be either numeric parameters, such as line lengths or circle diameters, or geometric parameters, such as tangent, parallel, concentric, horizontal or vertical, etc. Numeric parameters can be associated with each other through the use of relations, which allows them to capture design intent.

Design intent is how the creator of the part wants it to respond to changes and updates. For example, you would want the hole at the top of a beverage can to stay at the top surface, regardless of the height or size of the can. SolidWorks allows the user to specify that the hole is a feature on the top surface, and will then honor their design intent no matter what height they later assign to the can.

Features refer to the building blocks of the part. They are the shapes and operations that construct the part. Shape-based features typically begin with a 2D or 3D sketch of shapes such as bosses, holes, slots, etc. This shape is then extruded to add or cut to remove material from the part. Operation-based features are not sketch-based, and include features such as fillets, chamfers, shells, applying draft to the faces of a part, etc.

3.1.2. *Computer-Aided Design (CAD)*

3.1.2.1. Base Design

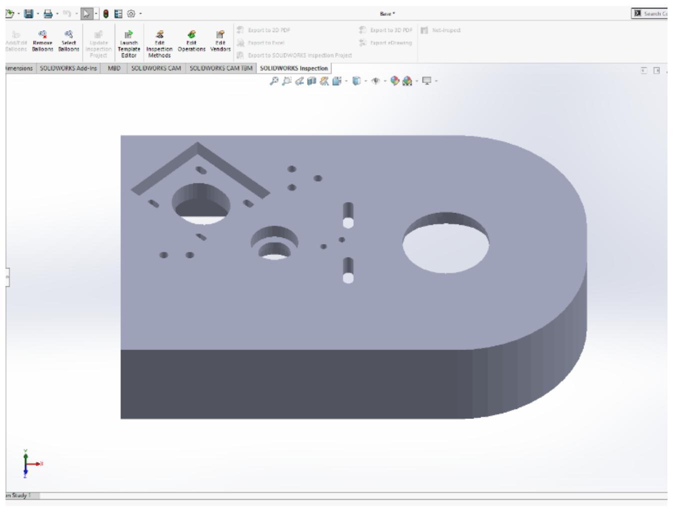


FIGURE 15

Base Design

3.1.2.2. Base Cover

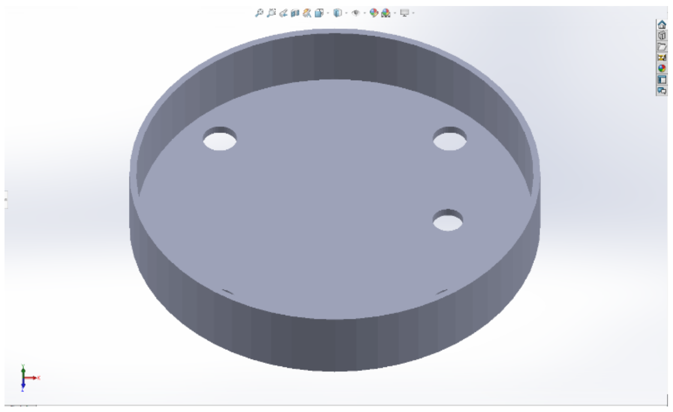


FIGURE 16

Base Cover

3.1.2.3. Base Assembling

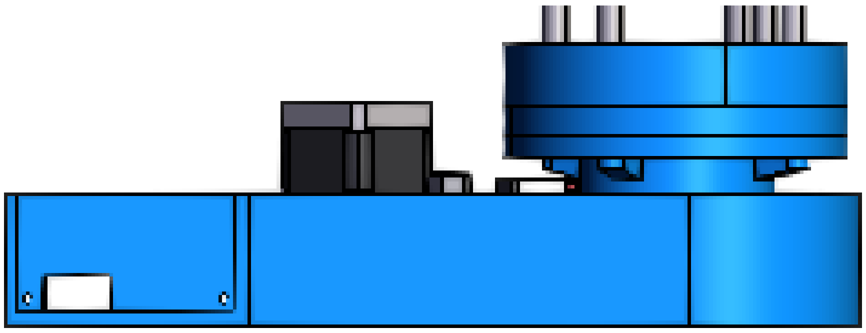


FIGURE 17

Base Assembling

3.1.2.4. Arm 1

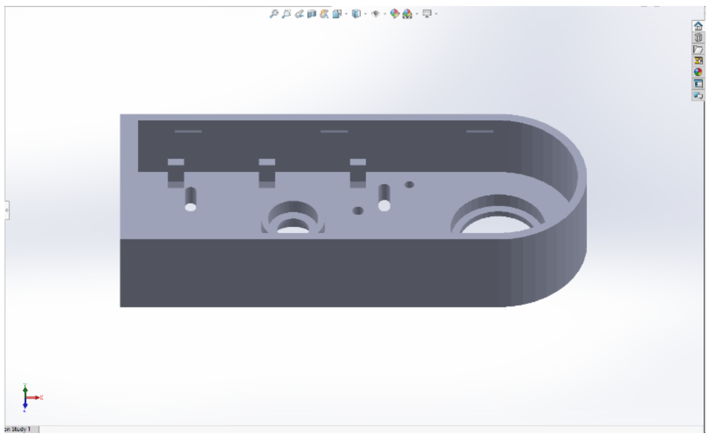


FIGURE 18

Base Arm

3.1.2.5. Arm 1 Cover

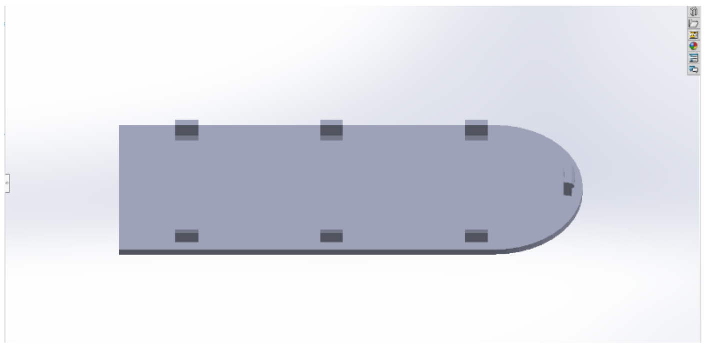


FIGURE 19

Arm 1 Cover

3.1.2.6. Arm 2

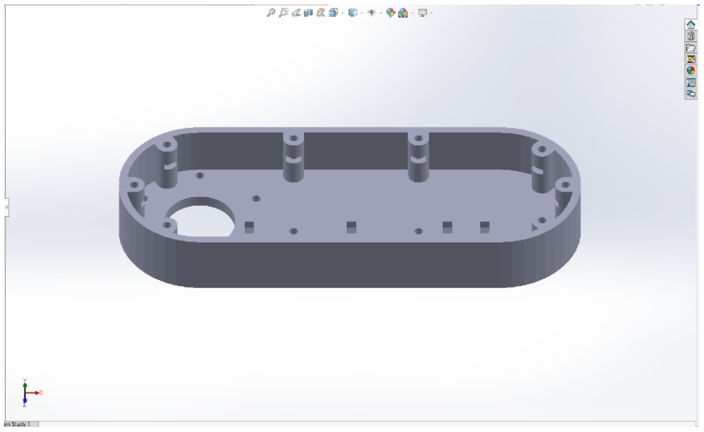


FIGURE 20

Arm 2

3.1.2.7. Arm 2 Cover

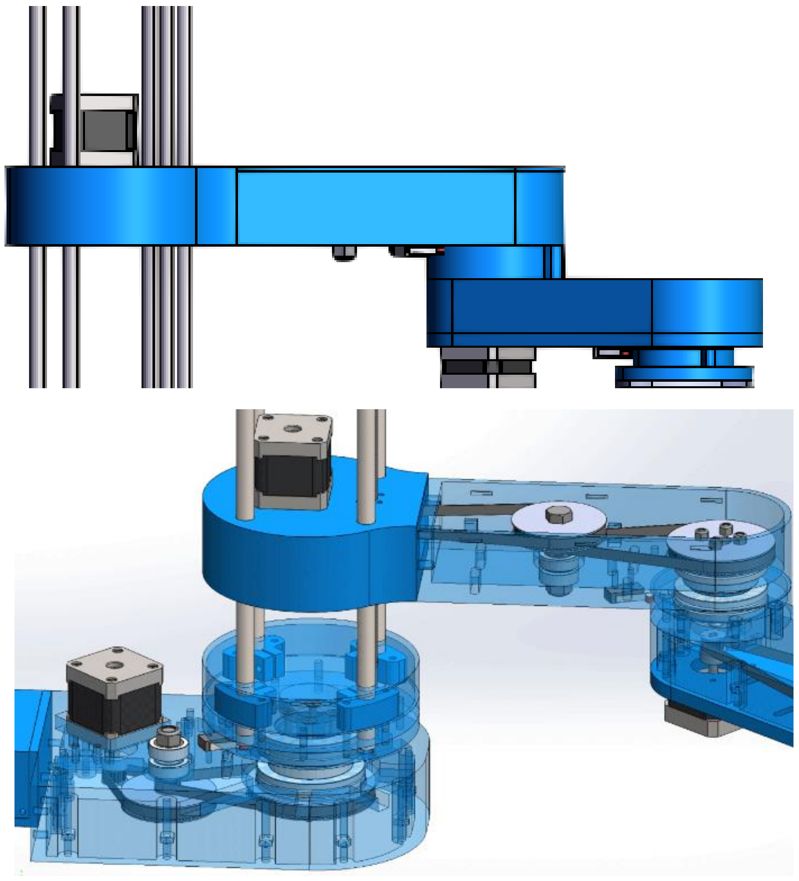


FIGURE 21

Arm 2 Cover

3.1.2.8. Gripper Mechanism

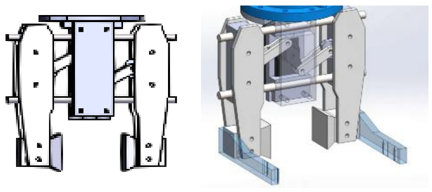


FIGURE 22

Gripper Mechanism

3.2.9. Motors, Axis, Pulley, Couplers Connecting

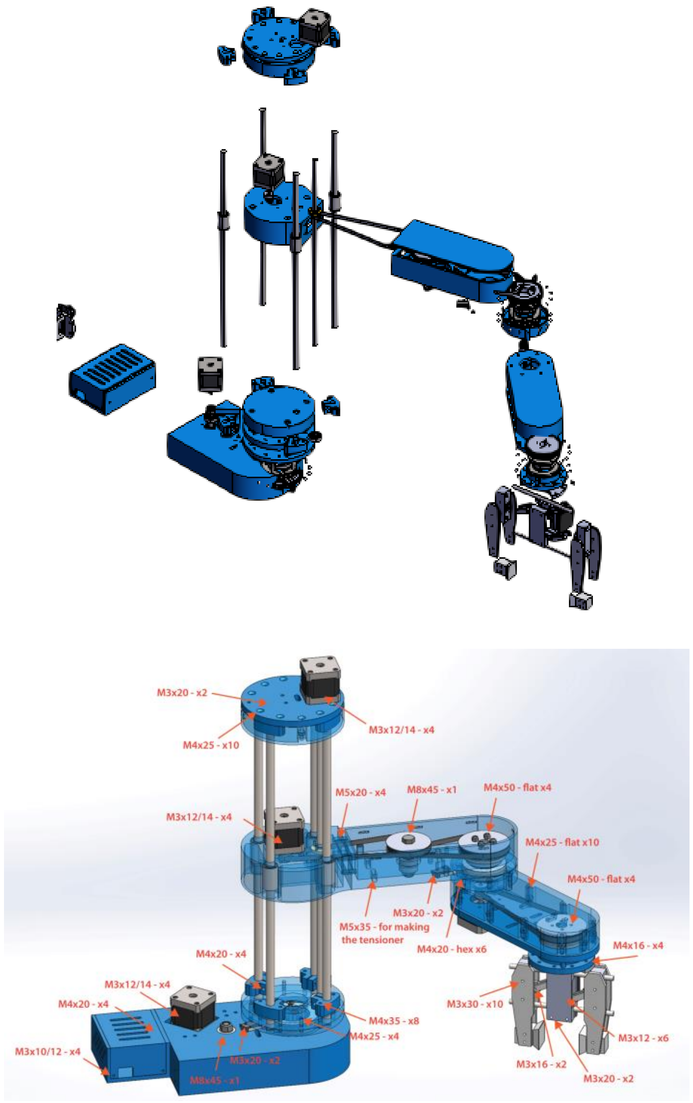
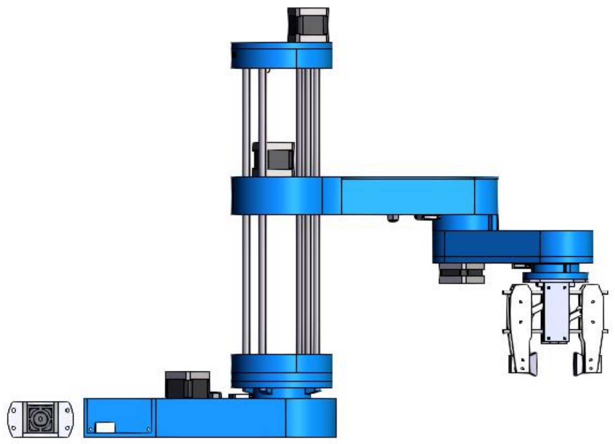


FIGURE 23

Motors, Axis, Pully, Cuplers Connecting

3.1.2.10. Robot Assembling



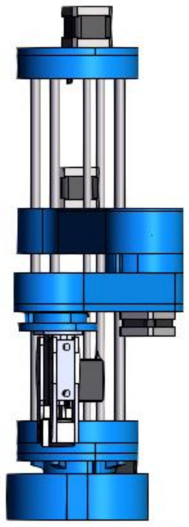


FIGURE 24

Robot Assembling

3.1.3. Components

- 4x Smooth rod shaft – 10mm 400mm
- 1x Lead screw – 8mm 400mm
- 4x Linear bearings 10mm
- 1x Thrust ball bearing 40x60x13mm
- 2x Thrust ball bearing 35x52x12mm
- 5x Radial ball bearing 8x22x7mm
- Various lengths M3, M4 and M5 bolts and nuts

3.4. 3D Printing and Assembling

3D printing, also known as additive manufacturing, is a process in which a digital file is used to create a three-dimensional solid object. In the 3D printing process, sequential layers of material are laid down by the ‘3D printer’ until object creation is completed.

3D-printed objects are created through an additive process, where the printer places layer after layer of material until the desired thing is ‘printed’. Each layer can be considered a finely sliced cross-section of the printed item. With 3D printing, users can produce complicated shapes without consuming as much material as traditional manufacturing methods require.

The operation style of 3D printing is the opposite of ‘subtractive manufacturing’, where the material is cut out or hollowed using equipment such as a milling machine. Conversely, additive manufacturing does not need a mold or material block to create physical objects. Instead, it stacks layers of material and fuses them together.

3D printing offers swift product creation, low expenses for the initial fixed infrastructure, and the ability to create complicated geometries using several material types, something traditional manufacturing solutions might not be capable of as efficiently.

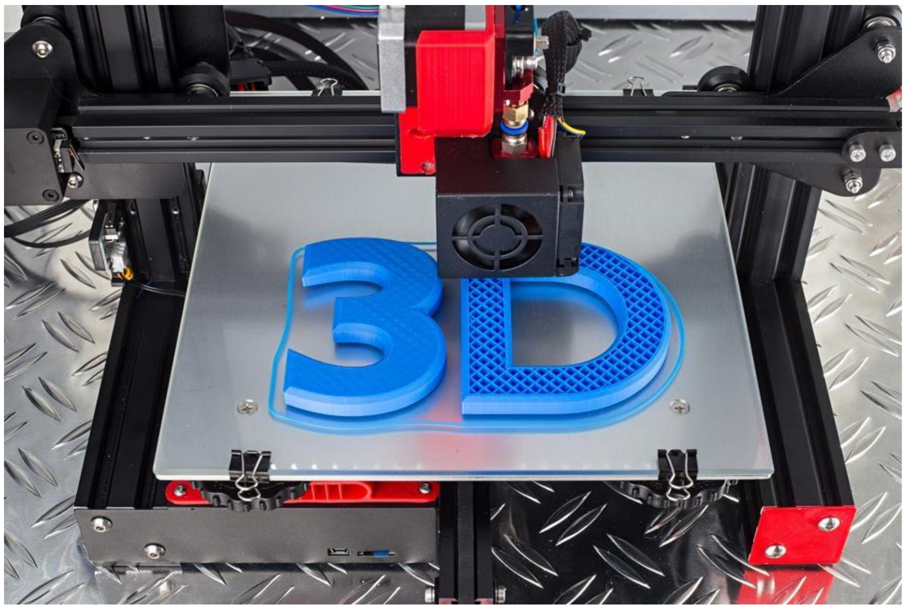


FIGURE 25

3D Printer

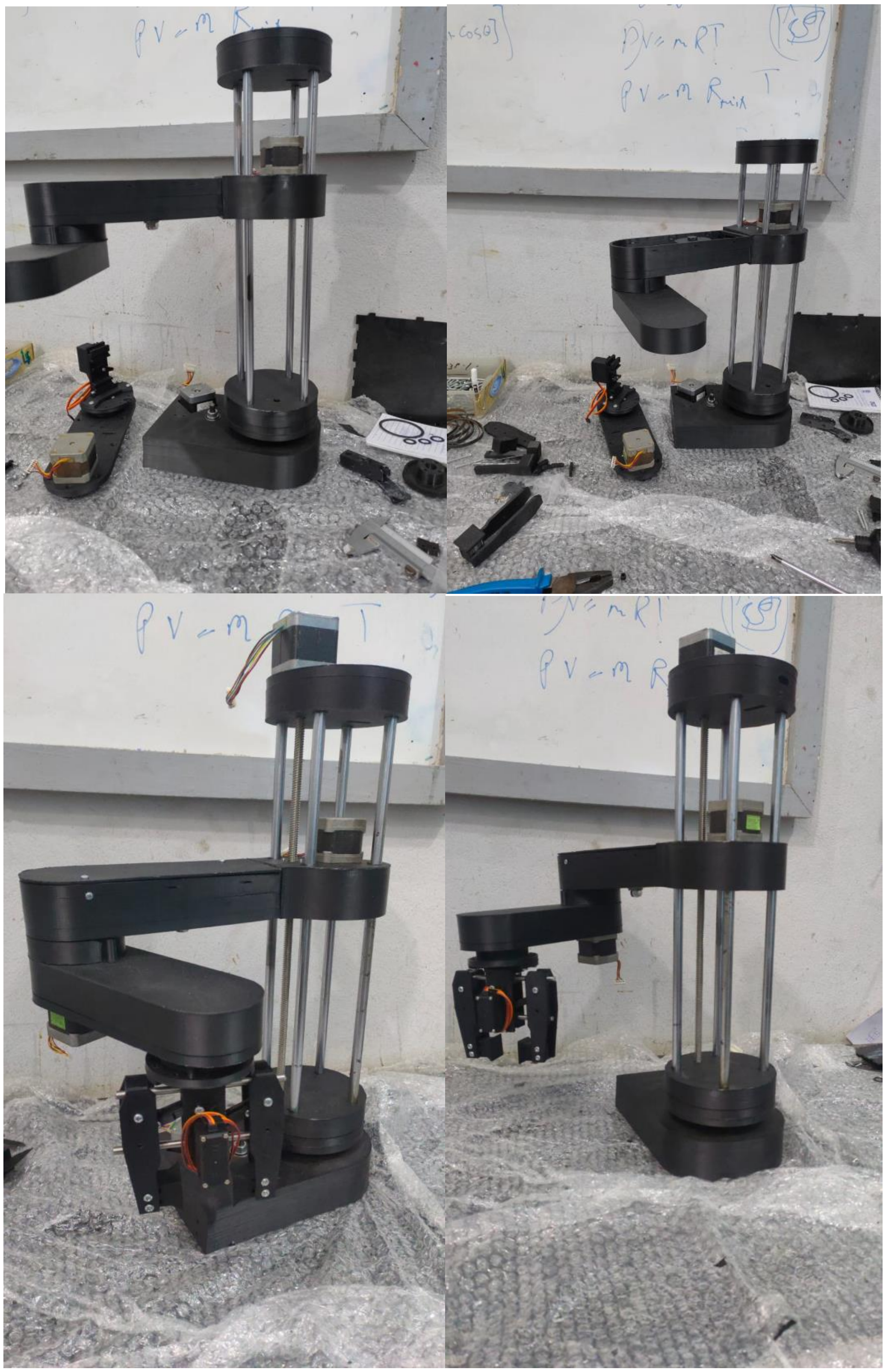


FIGURE 26

Our Robot

CHAPTER 2: HARDWARE

3.2. Electrical Design

3.2.1. Project components:

- Arduino Uno
- DC Power Supply
- Arduino CNC Shield
- Stepper Motor – NEMA 17
- A4988 Stepper Driver
- Limit Switch

3.2.1.1. Arduino Uno

One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web that can help you get started or out of a jam.

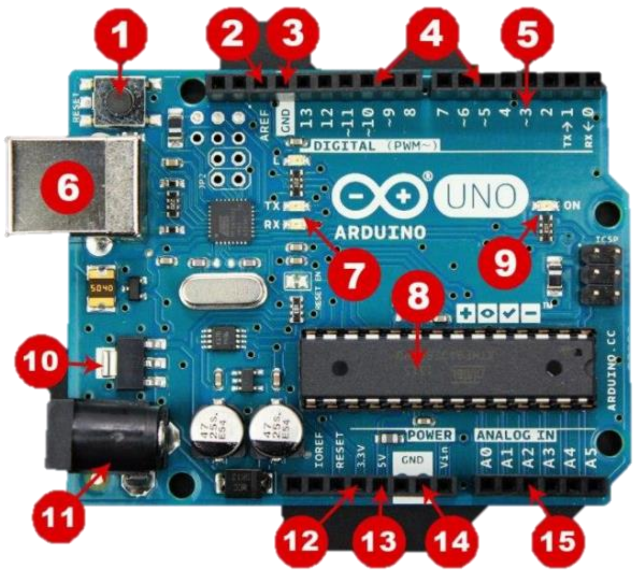


FIGURE 27

Arduino Uno Board Breakdown

Board Breakdown

Here are the components that make up an Arduino board and what each of their functions are.

1. Reset Button – This will restart any code that is loaded to the board.
2. AREF – Stands for "Analog Reference" and is used to set an external reference voltage

3. Ground Pin – There are a few ground pins on the Arduino and they all work the same
4. Digital Input/Output – Pins 0-13 can be used for digital input or output
5. PWM – The pins marked with the (~) symbol can simulate analog output
6. USB Connection – Used for powering up your Arduino and uploading sketches
7. TX/RX – Transmit and receive data indication LEDs
8. ATmega Microcontroller – This is the brains and is where the programs are stored
9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source
10. Voltage Regulator – This controls the amount of voltage going into the Arduino board
11. DC Power Barrel Jack – This is used for powering your Arduino with a power supply
12. 3.3V Pin – This pin supplies 3.3 volts of power to your projects
13. 5V Pin – This pin supplies 5 volts of power to your projects
14. Ground Pins – There are a few ground pins on the Arduino and they all work the same
15. Analog Pins – These pins can read the signal from an analog sensor and convert it to digital

3.2.1.2. DC Power Supply

The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. You can do what most people do and connect the board directly to your computer via a USB cable. If you want your project to be mobile, consider using a 9V battery pack to give it juice. The last method would be to use a 9V AC power supply.

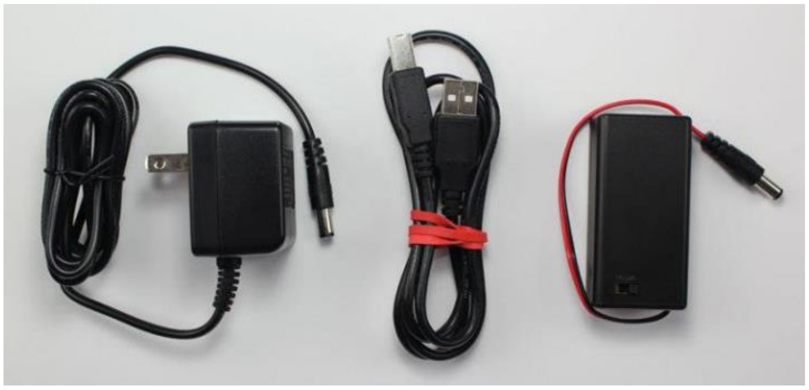


FIGURE 28

Arduino Power Supply

3.2.1.3. Arduino CNC Shield

The CNC Shield V3 for Arduino®, is an Arduino compatible board that turns your Arduino into a CNC controller. Using an opensource firmware it can control up to 4 Stepper motor using DRV8825 or A4988 stepper motor driver making it easy to get your CNC projects up and running in a few hours.

Features:

- GRBL 0.8c compatible. (Open source firmware that runs on an Arduino UNO that turns G-code commands into stepper signals https://github.com/grbl/grbl)
- 4-Axis support (X, Y, Z , A-Can duplicate X,Y,Z or do a full 4th axis with custom firmware using pins D12 and D13)
- 2 x End stops for each axis (6 in total)
- Spindle enable and direction
- Coolant enable
- Uses removable Pololu A4988 compatible stepper drivers. (A4988, DRV8825 and others)
- Jumpers to set the Micro-Stepping for the stepper drivers. (Some drivers like the DRV8825 can do up to 1/32 micro-stepping)
- Compact design.
- Stepper Motors can be connected with 4 pin molex connectors.
- Runs on 12-36V DC. (At the moment only the Pololu DRV8825 drivers can handle up to 36V so please consider the operation voltage when powering the board.)

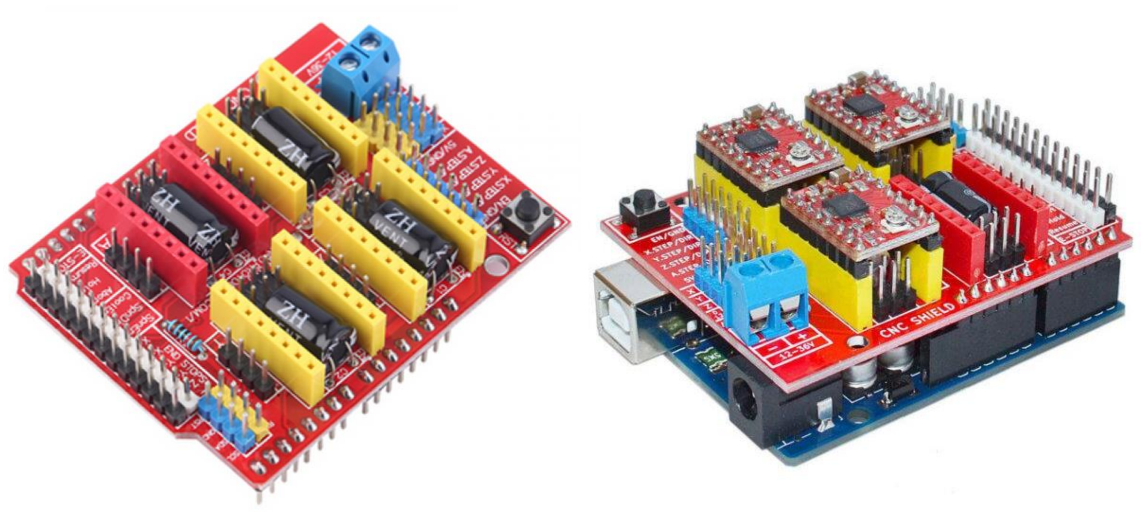


FIGURE 29

CNC Shield V3

3.2.1.4. Stepper Motor - NEMA 17

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation.Its normal shaft motion consists of discrete angular movements of essentially uniform magnitude when driven from sequentially switched DC power supply.

The stepper motor is a digital input-output device. It is particularly well suited to the type of application where control signals appear as digital pulses rather than analog voltages. One digital pulse to a stepper motor drive or translator causes the motor to increment one precise angle of motion. As the digital pulses increase in frequency, the step movement changes into continuous rotation.

Some industrial and scientific applications of stepper motors include robotics, machine tools, pick and place machines, automated wire cutting and wire bonding machines, and even precise fluid control devices.

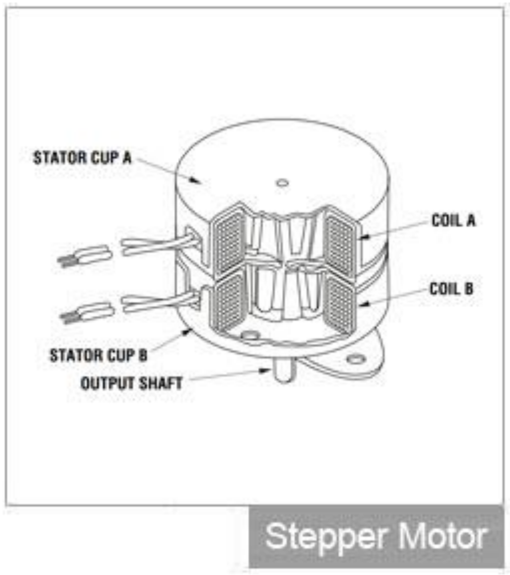


FIGURE 30

Stepper Motor

NEMA 17 is a **hybrid stepping motor** with a 1.8° step angle (200 steps/revolution). Each phase draws 1.2 A at 4 V, allowing for a holding torque of 3.2 kg-cm. NEMA 17 Stepper motor is generally used in Printers, CNC machines and Laser Cutters.

NEMA 17 Stepper Motor Technical Specifications

- Rated Voltage: 12V DC
- Current: 1.2A at 4V
- Step Angle: 1.8 deg.

- No. of Phases: 4
- Motor Length: 1.54 inches
- 4-wire, 8 inch lead
- 200 steps per revolution, 1.8 degrees
- Operating Temperature: -10 to 40 °C
- Unipolar Holding Torque: 22.2 oz-in

Other Stepper motors
Nema23, Nema34, 28BYJ-48 Stepper Motor

Other Motors
DC Motor, 12V DC motor, Servo Motor, BLDC Motor

Where to use NEMA 17 Stepper Motor

NEMA17 Stepper Motor is commonly used in CNC machines, Hard Drives and Linear Actuators. The motor have 6 lead wires and rated voltage is 12 volt. It can be operated at lower voltage but torque will drop.

How to use 28-BYJ48 Stepper Motor

These stepper motors consume high current and hence a driver IC like the A4988 is mandatory. To know how to make this motor rotate we should look into the coil diagram below.

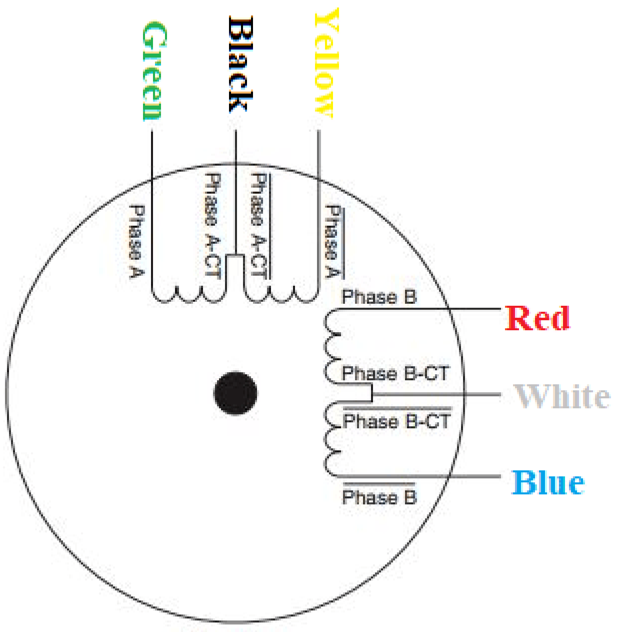


FIGURE 31

How to use 28-BYJ48 Stepper Motor

The motor has six wires, connected to two split windings as is common for unipolar stepper motors. Black, Yellow, Green wires is part of first winding where Black is centre tap and Yellow and Green are coil end while Red, White and Blue is part of second winding in which White is centre tap and Red and Blue are coil end wires. In use, the centre taps of the windings (Black and White) are typically wired to the positive supply, and the two ends of each winding are alternately grounded through a drive circuit. As shown in the wiring diagram the order of the stator poles in the motor is A, B, A', B'.

Stepper Motor Applications

- CNC machines
- Precise control machines
- 3D printer/prototyping machines (e.g. RepRap)
- Laser cutters
- Pick and place machines

NEMA17 Dimensions

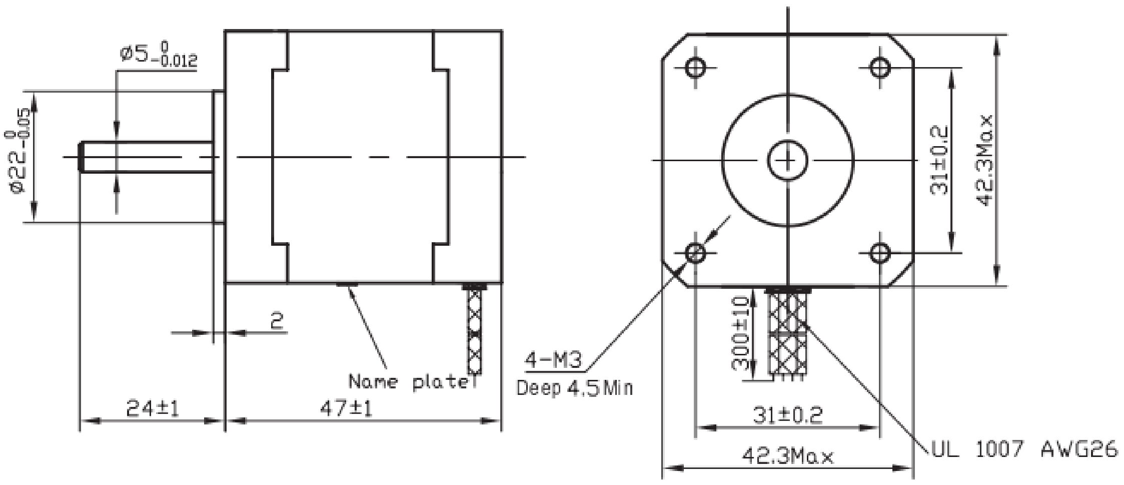


FIGURE 32

NEMA17 Dimensions

3.2.1.5. A4988 Stepper Driver

For single-stepper-motor applications, a driver like the L298N is fine, but if you want to construct your own CNC machine or 3D printer, you'll need a dedicated stepper motor driver like the A4988.

Due to the simplicity of the step motor control and the variety of stepping modes provided by the A4988 driver, it is an ideal solution for building applications that require precise and reliable stepper motor control, such as the movement control of beds, heads, and assemblies in various CNC plotting, milling, and 3D printer designs.

The fact that it only requires two pins to control the speed and direction of a bipolar stepper motor like the NEMA 17 is pretty neat, too.

A4988 Stepper Motor Driver Chip

At the heart of the module is a microstepping driver from Allegro – A4988. Despite its small stature (0.8″x0.6″), it packs quite a punch.

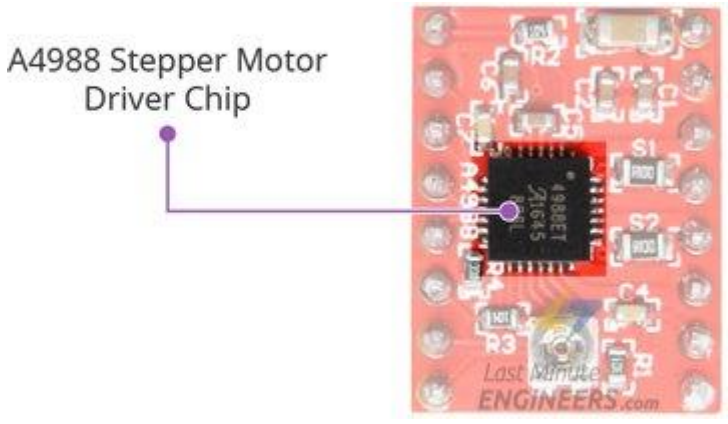


FIGURE 33

A4988 Stepper Motor Driver Chip

The A4988 stepper motor driver has an output drive capacity of up to 35V and ±2A. This allows you to control a bipolar stepper motor, such as the NEMA 17, at up to 2A output current per coil.

Furthermore, the output current is regulated, allowing for noiseless operation of the stepper motor and the elimination of resonance or ringing that is common in unregulated stepper driver designs.

The driver has a built-in translator for easy operation. This reduces the number of control pins to just two, one for controlling the steps and the other for controlling the spinning direction.

The driver offers five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step.

In order to ensure reliable operation, the driver has additional features such as under-voltage, shoot-through, short circuit, overcurrent, and thermal protection.

Technical Specifications

Here are complete specifications:

Motor output voltage	8V – 35V
Logic input voltage	3V – 5.5V
Continuous current per phase	1A
Maximum current per phase	2A
Microstep resolution	full, 1/2, 1/4, 1/8 and 1/16

For more information, please refer to the datasheet below.

A4988 Datasheet

A4988 Motor Driver Pinout

The A4988 driver has a total of 16 pins that connect it to the outside world. The pinout is as follows:

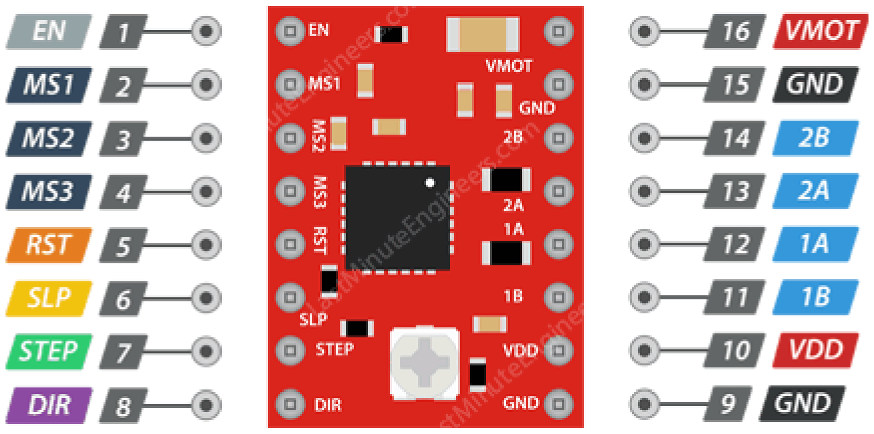


FIGURE 34

A4988 Motor Driver Pinout

Let’s get to know all the pins one by one.

Power Pins

The A4988 actually requires two power supply connections.

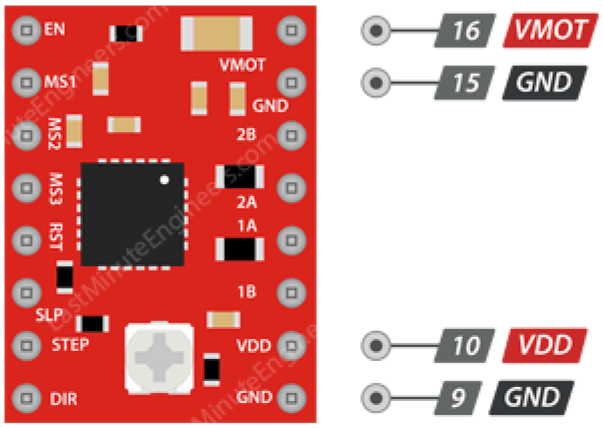


FIGURE 35

Power Pins

VDD and GND are used to power the internal logic circuitry, which can range from 3V to 5.5V.

Whereas,

VMOT and GND supply power to the motor, which can range from 8V to 35V.

According to the datasheet, in order to sustain 4A, the motor supply requires a suitable decoupling capacitor close to the board.

Microstep Selection Pins

The A4988 driver supports microstepping by dividing a single step into smaller steps. This is achieved by energizing the coils with intermediate current levels. For example, if you choose to drive the NEMA 17 (with 1.8° step angle or 200 steps/revolution) in quarter-step mode, the motor will produce 800 microsteps per revolution.

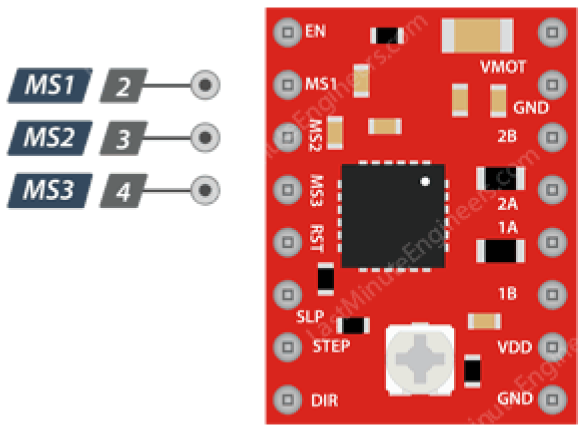


FIGURE 36

Microstep Selection Pins

The A4988 driver has three step size (resolution) selector inputs: MS1, MS2 & MS3. By setting the appropriate logic levels for these pins, we can set the motor to one of five step resolutions.

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

These three microstep selection pins are pulled LOW by internal pull-down resistors, so if you leave them unconnected, the motor will operate in full step mode.

Control Input Pins

The A4988 has two control inputs: STEP and DIR.

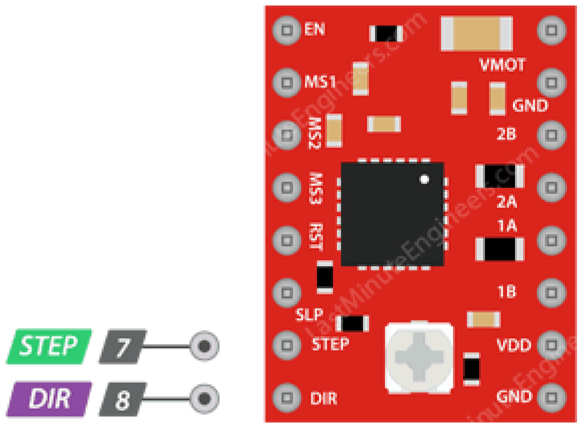


FIGURE 37

Control Input Pins

STEP input controls the microsteps of the motor. Each HIGH pulse sent to this pin drives the motor according to the number of microsteps determined by the microstep selection pins. The higher the pulse frequency, the faster the motor will spin.

DIR input controls the spinning direction of the motor. Pulling it HIGH turns the motor clockwise, while pulling it LOW turns it counterclockwise.

If you want the motor to only turn in one direction, you can connect the DIR directly to VCC or GND.

Pins For Controlling Power States

The A4988 has three separate inputs for controlling its power states: EN, RST, and SLP.

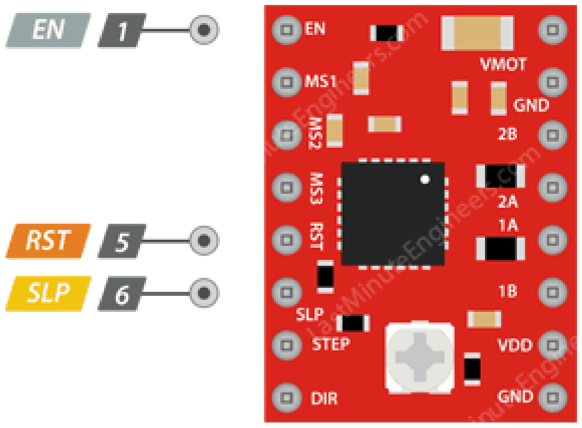


FIGURE 38

Pins for Controlling Power States

EN is an active low input pin. When this pin is pulled LOW, the A4988 driver is enabled. By default, this pin is pulled low, so unless you pull it high, the driver is always enabled. This pin is particularly useful when implementing an emergency stop or shutdown system.

SLP is an active low input pin. Pulling this pin LOW puts the driver into sleep mode, reducing power consumption to a minimum. You can use this to save power, especially when the motor is not in use.

RST is an active low input as well. When this pin is pulled LOW, all STEP inputs are ignored. It also resets the driver by setting the internal translator to a predefined "home" state. Home state is basically the initial position from which the motor starts, and it varies based on microstep resolution.

Output Pins

The output channels of the A4988 motor driver are broken out to the side of the module with pins 1B, 1A, 2A & 2B.

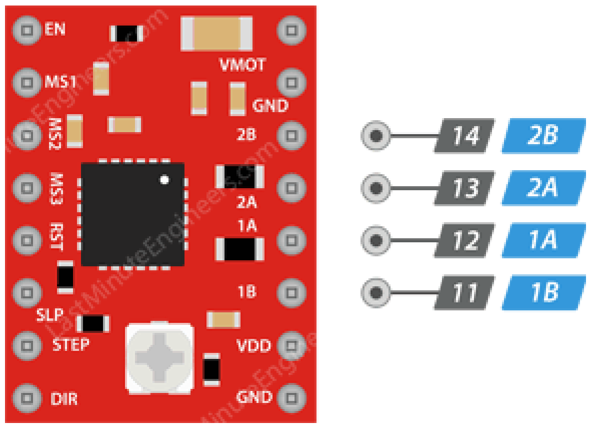


FIGURE 39

Output Pins

You can connect any small to medium-sized bipolar stepper motor, such as NEMA 17, to these pins.

Each output pin can supply up to 2A to the motor. However, the amount of current supplied to the motor is determined by the power supply, cooling system, and current limiting setting of the system.

Cooling System – Heatsink

Excessive power dissipation of the A4988 driver IC causes a temperature rise, which could potentially damage the IC if it exceeds its capacity.

Despite having a maximum current rating of 2A per coil, the A4988 driver IC can only supply about 1A per coil without overheating. To achieve more than 1A per coil, a heat sink or other cooling method is required.

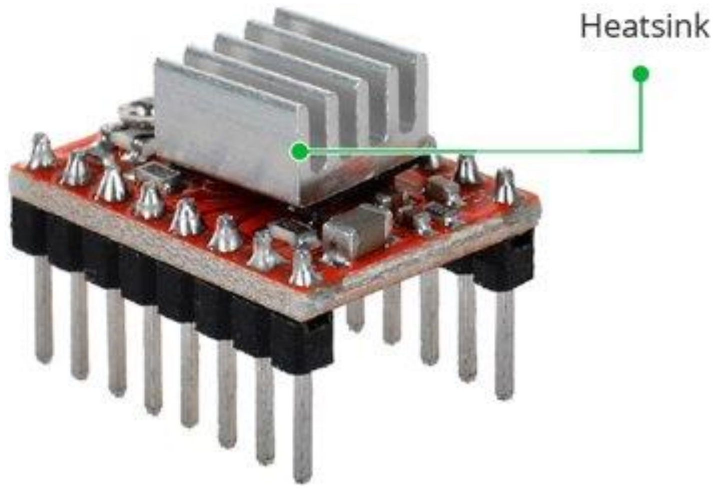


FIGURE 40

Cooling System – Heatsink

Current limiting

Before running the motor, you must limit the maximum current flowing through the stepper coils so that it does not exceed the motor's rated current.

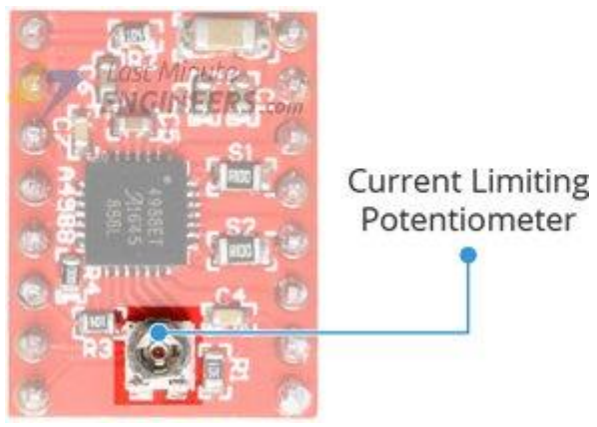


FIGURE 41

Current limiting

The A4988 driver includes a small trimmer potentiometer for setting the current limit.

There are two methods for making this adjustment:

Method 1:

In this method, the current limit is determined by measuring the voltage (Vref) at the “ref” pin.

1. Take a look at the datasheet for your stepper motor. Make a note of the rated current. In our case, NEMA 17 200steps/rev, 12V 350mA is used.
2. Disconnect the three microstep selection pins to put the driver in full-step mode.
3. Hold the motor in a fixed position without clocking the STEP input.
4. Measure the voltage (Vref) on the metal trimmer pot as you adjust it.
5. Adjust the Vref voltage by using the formula

   Vref = Current Limit / 2.5
   For example, if your motor is rated at 350mA, you would set the reference voltage to 0.14V.

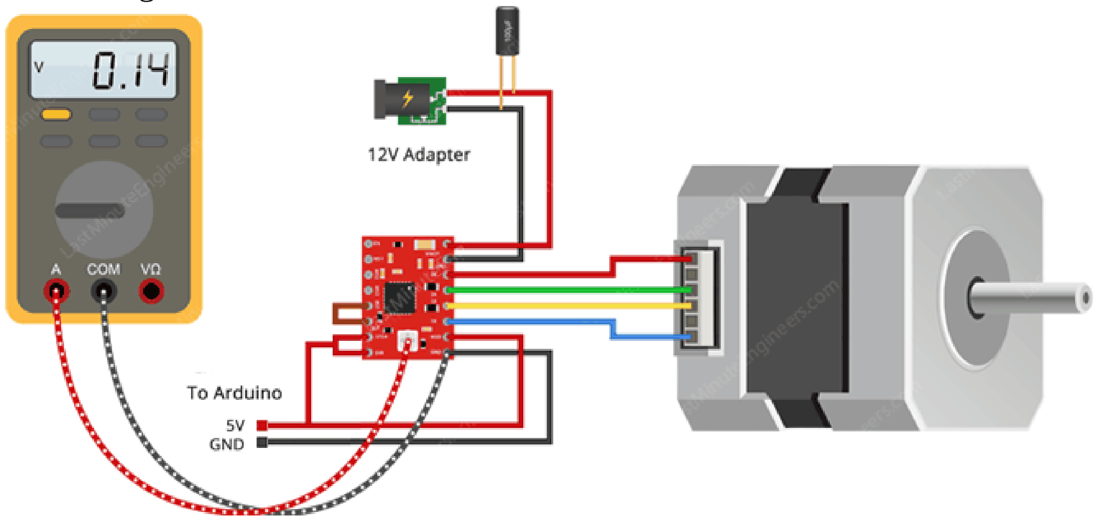


FIGURE 42

Adjustment Method 1

Method 2:

In this method, the current limit is determined by measuring the current flowing through the coil.

1. Take a look at the datasheet for your stepper motor. Make a note of the rated current. In our case, NEMA 17 200steps/rev, 12V 350mA is used.

2. Disconnect the three microstep selection pins to put the driver in full-step mode.

3. Hold the motor in a fixed position without clocking the STEP input. Don't leave the STEP input floating; instead, connect it to a logic power supply (5V).

4. Put the ammeter in series with one of the coils on your stepper motor and measure the actual current flowing.

5. Take a small screwdriver and adjust the current limit potentiometer until you reach the rated current.

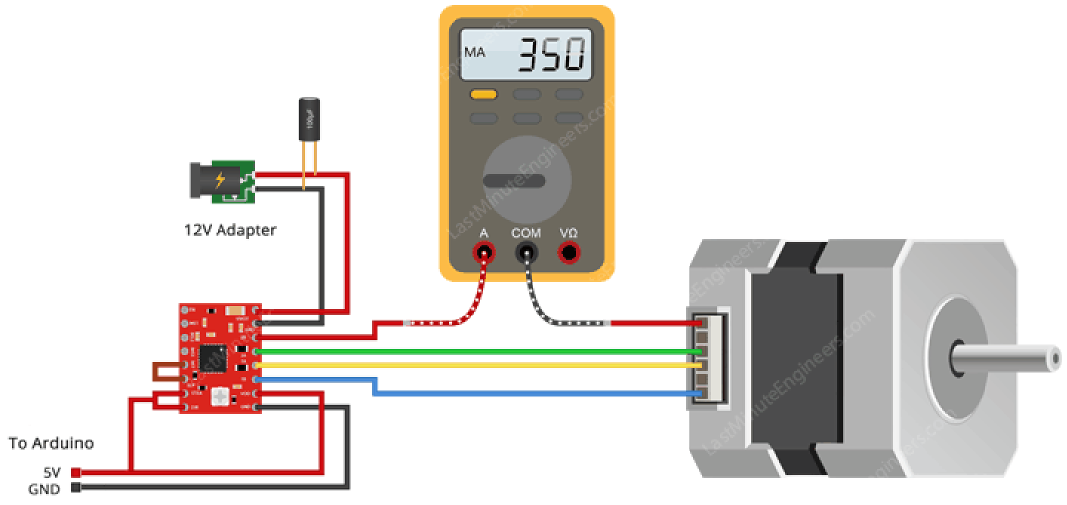


FIGURE 43

Adjustment Method 2

Wiring an A4988 Stepper Motor Driver to an Arduino

Now that we know everything about the driver, let's hook it up to our Arduino. The connections are straightforward. Begin by connecting VDD and GND (next to VDD) to the Arduino's 5V and Ground pins. Connect the DIR and STEP input pins to the Arduino's digital output pins #2 and #3.

Connect the stepper motor to the 2B, 2A, 1A, and 1B pins. Actually, the A4988 module is conveniently laid out to match the 4-pin connector on bipolar stepper motors, so that shouldn’t be a problem.

To keep the driver enabled, connect the RST pin to the adjacent SLP/SLEEP pin. Keep the microstep selection pins disconnected if you want to run the motor in full step mode.

Finally, connect the motor power supply to the VMOT and GND pins. Remember to put a large 100µF decoupling electrolytic capacitor across the motor power supply pins to avoid large voltage spikes.

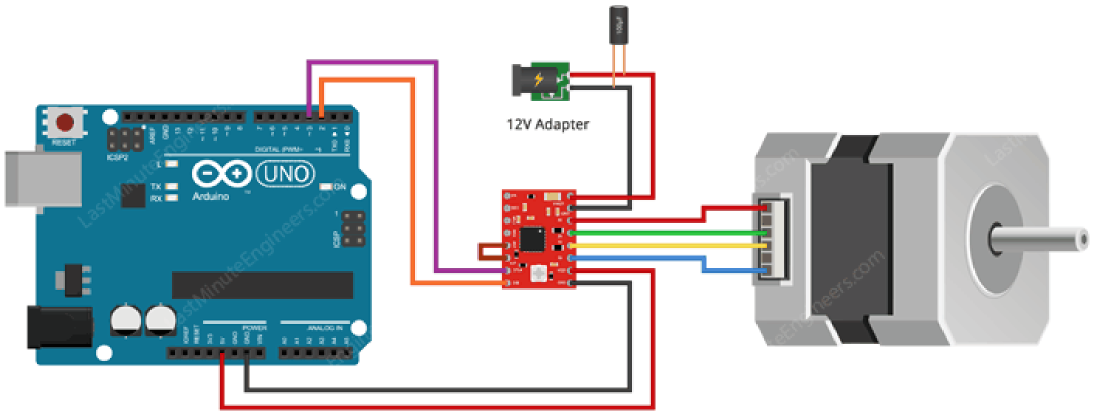


FIGURE 44

Wiring an A4988 Stepper Motor Driver to an Arduino

3.2.1.6. Limit Switch

A limit switch is a switch operated by the motion of a machine part or the presence of an object. A limit switch can be used for controlling machinery as part of a control system, as a safety interlock, or as a counter enumerating objects passing a point.[1]

Limit switches are used in a variety of applications and environments because of their ruggedness, ease of installation, and reliability of operation. They can determine the presence, passing, positioning, and end of travel of an object. They were first used to define the limit of travel of an object, hence the name "limit switch".

Standardized limit switches are industrial control components manufactured with a variety of operator types, including lever, roller plunger, and whisker type. Limit switches may be directly mechanically operated by the motion of the operating lever. A reed switch may be used to indicate proximity of a magnet mounted on some moving part. Proximity switches operate by the disturbance of an electromagnetic field, by capacitance, or by sensing a magnetic field.

Rarely, a final operating device such as a lamp or solenoid valve is directly controlled by the contacts of an industrial limit switch, but more typically the limit switch is wired through a control relay, a motor contactor control circuit, or as an input to a programmable logic controller.

FIGURE 45

Limit Switch

3.2.2. Project Circuit Diagram

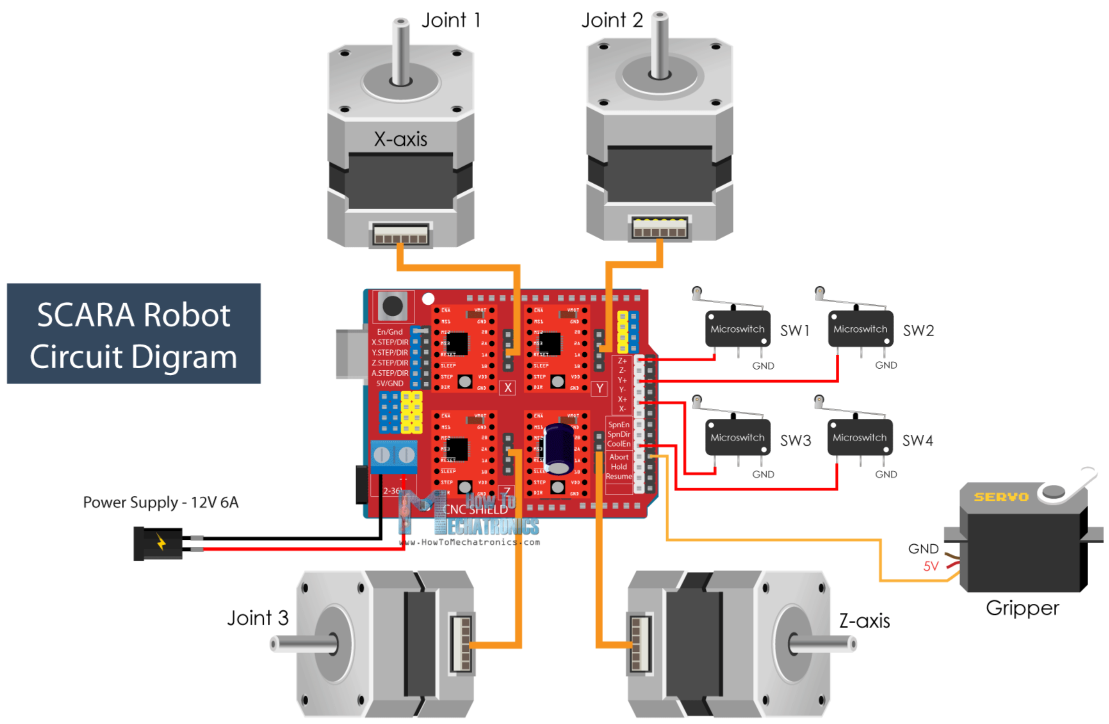


FIGURE 46

Circuit Diagram

CHAPTER 3: SOFTWARE

3.3. Software Design

Software design is the process of envisioning and defining software solutions to one or more sets of problems. One of the main components of software design is the software requirements analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering.

3.3.1. How the SCARA robot works?

There are two methods for controlling robots in terms of positioning and orientation, and that's using forward or inverse kinematics.

Forward kinematics is used when we need to find the position and orientation of the end-effector from the given joint angles.

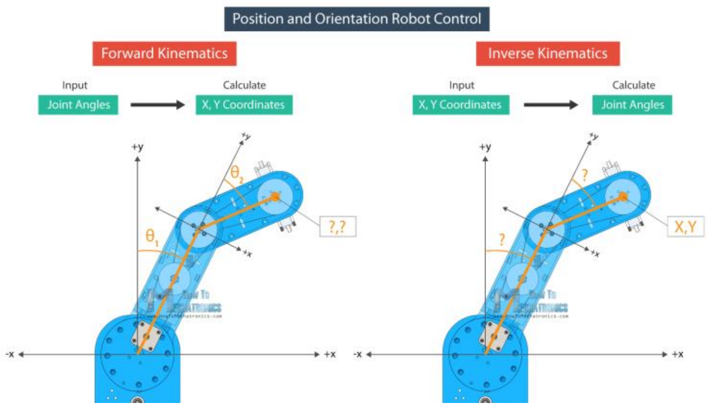


FIGURE 47

SCARA Forward kinematics

On the other hand, inverse kinematics is used when we need to find the joint angles for a given position of the end-effector. This method makes more sense in robotics as most of the time we want the robot to position its tool to a particular location or particular X, Y and Z coordinates.

With inverse kinematics we can calculate the joint angles according to given coordinates.

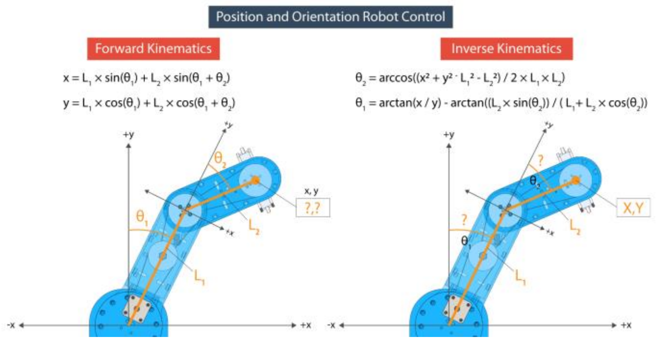
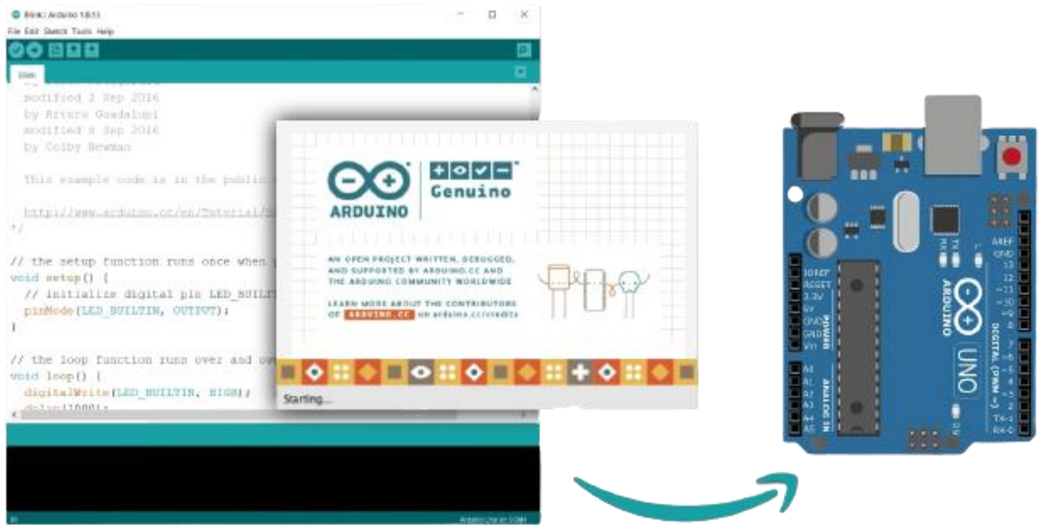

Position and Orientation Robot Control
Forward Kinematics
$x = L_1 \times \sin(\theta_1) + L_2 \times \sin(\theta_1 + \theta_2)$
$y = L_1 \times \cos(\theta_1) + L_2 \times \cos(\theta_1 + \theta_2)$
Inverse Kinematics
$\theta_2 = \arccos((x^2 + y^2 - L_1^2 - L_2^2) / 2 \times L_1 \times L_2)$
$\theta_1 = \arctan(x / y) - \arctan((L_2 \times \sin(\theta_2)) / (L_1 + L_2 \times \cos(\theta_2))$
+y
+x
-x
θ_1
θ_2
L_1
L_2
x, y
?,?
?
X,Y

FIGURE 48

SCARA Inverse Kinematics

3.3.2 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.


Blink | Arduino 1.8.1
File Edit Sketch Tools Help
Blink
ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits
Starting...
ARDUINO
UNO
POWER
ANALOG IN
DIGITAL (PWM ~)

```
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the publi

  http://www.arduino.cc/en/Tutorial/
*/

// the setup function runs once when
void setup() {
  // initialize digital pin LED_BUIL
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ov
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
```

FIGURE 49

Arduino Integrated Development Environment (IDE)

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

3.3.3. Program Code

We wrote the following code lines to develop our robot program:

3.3.3.1 Forward Kinematics Code

```
// FORWARD KINEMATICS

void forwardKinematics() {

 float theta1F = theta1 * PI / 180;   // degrees to radians

 float theta2F = theta2 * PI / 180;

 xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));

 yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));

}
```

3.3.3.2. Inverse Kinematics Code

```
/ INVERSE KINEMATICS

void inverseKinematics(float x, float y) {

 theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));

 if (x < 0 & y < 0) {

   theta2 = (-1) * theta2;

 }

 theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

 theta2 = (-1) * theta2 * 180 / PI;

 theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y
is

 if (x >= 0 & y >= 0) {       // 1st quadrant
```

```
  theta1 = 90 - theta1;
}
if (x < 0 & y > 0) {      // 2nd quadrant
  theta1 = 90 - theta1;
}
if (x < 0 & y < 0) {      // 3d quadrant
  theta1 = 270 - theta1;
  phi = 270 - theta1 - theta2;
  phi = (-1) * phi;
}
if (x > 0 & y < 0) {      // 4th quadrant
  theta1 = -90 - theta1;
}
if (x < 0 & y == 0) {
  theta1 = 270 + theta1;
}
// Calculate "phi" angle so gripper is parallel to the X axis
phi = 90 + theta1 + theta2;
phi = (-1) * phi;

// Angle adjustment depending in which quadrant the final tool coordinate x,y
is
if (x < 0 & y < 0) {      // 3d quadrant
  phi = 270 - theta1 - theta2;
}
if (abs(phi) > 165) {
  phi = 180 + phi;
```

```
 }


 theta1=round(theta1);

 theta2=round(theta2);

 phi=round(phi);

 cp5.getController("j1Slider").setValue(theta1);

 cp5.getController("j2Slider").setValue(theta2);

 cp5.getController("j3Slider").setValue(phi);

 cp5.getController("zSlider").setValue(zP); }
```

3.3.3.3. Gripper Code

```
if (gripperValuePrevious != gripperValue) {

  if (activeIK == false) {    // Check whether the inverseKinematics mode is active, Executre Forward kinematics only if inverseKinematics mode is off or false

    gripperAdd = round(cp5.getController("gripperValue").getValue());

    gripperValue=gripperAdd+50;

    updateData();

    println(data);

    myPort.write(data);

  }

 }
```

3.3.3.4. IP/OP Code

```
public void updateData() {

 data = str(saveStatus)

  +","+str(runStatus)

  +","+str(round(cp5.getController("j1Slider").getValue()))

  +","+str(round(cp5.getController("j2Slider").getValue()))
  +","+str(round(cp5.getController("j3Slider").getValue()))

  +","+str(round(cp5.getController("zSlider").getValue()))

  +","+str(gripperValue)

  +","+str(speedSlider)

  +","+str(accelerationSlider);

}
```

3.3.3.5. Arduino Code

```
if (Serial.available()) {

  content = Serial.readString(); // Read the incomding data from Processing

  // Extract the data from the string and put into separate integer variables
(data[] array)

  for (int i = 0; i < 10; i++) {

    int index = content.indexOf(","); // locate the first ","

    data[i] = atol(content.substring(0, index).c_str()); //Extract the number from
start to the ","

    content = content.substring(index + 1); //Remove the number from the
string

  }

  /*

   data[0] - SAVE button status

   data[1] - RUN button status

   data[2] - Joint 1 angle

   data[3] - Joint 2 angle

   data[4] - Joint 3 angle

   data[5] - Z position

   data[6] - Gripper value

   data[7] - Speed value
```

```
    data[8] - Acceleration value
   */
```

3.3.3.6. Motors Code

```
stepper1.moveTo(stepper1Position);
  stepper2.moveTo(stepper2Position);
  stepper3.moveTo(stepper3Position);
  stepper4.moveTo(stepper4Position);

  while (stepper1.currentPosition() != stepper1Position ||
stepper2.currentPosition() != stepper2Position || stepper3.currentPosition() !=
stepper3Position || stepper4.currentPosition() != stepper4Position) {
    stepper1.run();
    stepper2.run();
    stepper3.run();
    stepper4.run(); }
```

3.3.4. Processing IDE

FIGURE 50

Processing IDE

Processing is a free graphical library and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. It also provides a graphical user interface for simplifying the compilation and execution stage.

3.3.4.1. GUI for SCARA robot

```
import processing.serial.*;
import controlP5.*;
import static processing.core.PApplet.*;

Serial myPort;
ControlP5 cp5; // controlP5 object

int j1Slider = 0;
int j2Slider = 0;
int j3Slider = 0;
int zSlider = 100;
int j1JogValue = 0;
int j2JogValue = 0;
int j3JogValue = 0;
int zJogValue = 0;
int speedSlider = 500;
int accelerationSlider = 500;
int gripperValue = 180;
int gripperAdd=180;
int positionsCounter = 0;


int saveStatus = 0;
int runStatus = 0;

int slider1Previous = 0;
int slider2Previous = 0;
int slider3Previous = 0;
int sliderzPrevious = 100;
int speedSliderPrevious = 500;
int accelerationSliderPrevious = 500;
int gripperValuePrevious = 100;

boolean activeIK = false;

int xP=365;
int yP=0;
int zP=100;
float L1 = 228; // L1 = 228mm
float L2 = 136.5; // L2 = 136.5mm
float theta1, theta2, phi, z;

String[] positions = new String[100];

String data;

void setup() {

  size(960, 800);
  //myPort = new Serial(this, "COM3", 115200);
  cp5 = new ControlP5(this);

  PFont pfont = createFont("Arial", 25, true); // use true/false for smooth/no-smooth
  ControlFont font = new ControlFont(pfont, 22);
  ControlFont font2 = new ControlFont(pfont, 25);

  //J1 controls
  cp5.addSlider("j1Slider")
    .setPosition(110, 190)
    .setSize(270, 30)
    .setRange(-90, 266) // Slider range, corresponds to Joint 1 or theta1 angle that the robot can
move to
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;
  cp5.addButton("j1JogMinus")
    .setPosition(110, 238)
    .setSize(90, 40)
    .setFont(font)
    .setCaptionLabel("JOG-")
    ;
  cp5.addButton("j1JogPlus")
    .setPosition(290, 238)
    .setSize(90, 40)
    .setFont(font)
    .setCaptionLabel("JOG+")
    ;
  cp5.addNumberbox("j1JogValue")
    .setPosition(220, 243)
    .setSize(50, 30)
    .setRange(0, 20)
    .setFont(font)
    .setMultiplier(0.1)
    .setValue(1)
    .setDirection(Controller.HORIZONTAL) // change the control direction to left/right
    .setCaptionLabel("")
    ;

  //J2 controls
  cp5.addSlider("j2Slider")
    .setPosition(110, 315)
    .setSize(270, 30)
    .setRange(-150, 150)
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;
  cp5.addButton("j2JogMinus")
    .setPosition(110, 363)
    .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG-")
  ;
cp5.addButton("j2JogPlus")
  .setPosition(290, 363)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG+")
  ;
cp5.addNumberbox("j2JogValue")
  .setPosition(220, 368)
  .setSize(50, 30)
  .setRange(0, 20)
  .setFont(font)
  .setMultiplier(0.1)
  .setValue(1)
  .setDirection(Controller.HORIZONTAL) // change the control direction to left/right
  .setCaptionLabel("")
  ;
//J3 controls
cp5.addSlider("j3Slider")
  .setPosition(110, 440)
  .setSize(270, 30)
  .setRange(-162, 162)
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("")
  ;
cp5.addButton("j3JogMinus")
  .setPosition(110, 493)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG-")
  ;
cp5.addButton("j3JogPlus")
  .setPosition(290, 493)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG+")
  ;
cp5.addNumberbox("j3JogValue")
  .setPosition(220, 493)
  .setSize(50, 30)
  .setRange(0, 20)
  .setFont(font)
  .setMultiplier(0.1)
  .setValue(1)
  .setDirection(Controller.HORIZONTAL) // change the control direction to left/right
  .setCaptionLabel("")
  ;

//Z controls
```

```
cp5.addSlider("zSlider")
  .setPosition(110, 565)
  .setSize(270, 30)
  .setRange(0, 150)
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("")
  ;
cp5.addButton("zJogMinus")
  .setPosition(110, 618)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG-")
  ;
cp5.addButton("zJogPlus")
  .setPosition(290, 618)
  .setSize(90, 40)
  .setFont(font)
  .setCaptionLabel("JOG+")
  ;
cp5.addNumberbox("zJogValue")
  .setPosition(220, 618)
  .setSize(50, 30)
  .setRange(0, 20)
  .setFont(font)
  .setMultiplier(0.1)
  .setValue(1)
  .setDirection(Controller.HORIZONTAL) // change the control direction to left/right
  .setCaptionLabel("")
  ;


cp5.addTextfield("xTextfield")
  .setPosition(530, 205)
  .setSize(70, 40)
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;
cp5.addTextfield("yTextfield")
  .setPosition(680, 205)
  .setSize(70, 40)
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;
cp5.addTextfield("zTextfield")
  .setPosition(830, 205)
  .setSize(70, 40)
  .setFont(font)
  .setColor(255)
  .setCaptionLabel("")
  ;

cp5.addButton("move")
  .setPosition(590, 315)
  .setSize(240, 45)
  .setFont(font)
  .setCaptionLabel("MOVE TO POSITION")
  ;

cp5.addButton("savePosition")
  .setPosition(470, 520)
  .setSize(215, 50)
  .setFont(font2)
  .setCaptionLabel("SAVE POSITION")
  ;

cp5.addButton("run")
  .setPosition(725, 520)
  .setSize(215, 50)
  .setFont(font2)
  .setCaptionLabel("RUN PROGRAM")
  ;

cp5.addButton("updateSA")
  .setPosition(760, 590)
  .setSize(150, 40)
  .setFont(font)
  .setCaptionLabel("(Update)")
  ;

cp5.addButton("clearSteps")
  .setPosition(490, 650)
  .setSize(135, 40)
  .setFont(font)
  .setCaptionLabel("(CLEAR)")
  ;

//Z controls
cp5.addSlider("speedSlider")
  .setPosition(490, 740)
  .setSize(180, 30)
  .setRange(500, 4000)
  .setColorLabel(#3269c2)
  .setFont(font)
  .setCaptionLabel("")
  ;

cp5.addSlider("accelerationSlider")
  .setPosition(720, 740)
  .setSize(180, 30)
  .setRange(500, 4000)
  .setColorLabel(#3269c2)
```

```
    .setFont(font)
    .setCaptionLabel("")
    ;
  cp5.addSlider("gripperValue")
    .setPosition(605, 445)
    .setSize(190, 30)
    .setRange(0, 100)
    .setColorLabel(#3269c2)
    .setFont(font)
    .setCaptionLabel("")
    ;
}

void draw() {
  background(#F2F2F2); // background black
  textSize(26);
  fill(33);
  text("Forward Kinematics", 120, 135);
  text("Inverse Kinematics", 590, 135);
  textSize(40);
  text("SCARA Robot Control", 260, 60);
  textSize(45);
  text("J1", 35, 250);
  text("J2", 35, 375);
  text("J3", 35, 500);
  text("Z", 35, 625);
  textSize(22);
  text("Speed", 545, 730);
  text("Acceleration", 745, 730);

  //println("PREV: "+accelerationSlider);
  fill(speedSlider);
  fill(accelerationSlider);
  fill(j1Slider);
  fill(j2Slider);
  fill(j3Slider);
  fill(zSlider);
  fill(j1JogValue);
  fill(j2JogValue);
  fill(j3JogValue);
  fill(zJogValue);
  fill(gripperValue);


  updateData();
  //println(data);

  saveStatus=0; // keep savePosition variable 0 or false. See, when button SAVE pressed it makes
the value 1, which indicates to store the value in the arduino code

  // If slider moved, calculate new position of X,Y and Z with forward kinematics
  if (slider1Previous != j1Slider) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre
Forward kinematics only if inverseKinematics mode is off or false
      theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
      theta2 = round(cp5.getController("j2Slider").getValue());
      forwardKinematics();
      myPort.write(data);
    }
  }
  slider1Previous = j1Slider;

  if (slider2Previous != j2Slider) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre
Forward kinematics only if inverseKinematics mode is off or false
      theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
      theta2 = round(cp5.getController("j2Slider").getValue());
      forwardKinematics();
      myPort.write(data);
    }
  }
  slider2Previous = j2Slider;

  if (slider3Previous != j3Slider) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre
Forward kinematics only if inverseKinematics mode is off or false
      theta1 = round(cp5.getController("j1Slider").getValue()); // get the value from the slider1
      theta2 = round(cp5.getController("j2Slider").getValue());
      forwardKinematics();
      myPort.write(data);
    }
  }
  slider3Previous = j3Slider;

  if (sliderzPrevious != zSlider) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre
Forward kinematics only if inverseKinematics mode is off or false
      zP = round(cp5.getController("zSlider").getValue());
      myPort.write(data);
    }
  }
  sliderzPrevious = zSlider;

  if (gripperValuePrevious != gripperValue) {
    if (activeIK == false) {     // Check whether the inverseKinematics mode is active, Executre
Forward kinematics only if inverseKinematics mode is off or false
      gripperAdd = round(cp5.getController("gripperValue").getValue());
      gripperValue=gripperAdd+50;
      updateData();
      println(data);
      myPort.write(data);
    }
  }
  gripperValuePrevious = gripperValue;
  activeIK = false; // deactivate inverseKinematics so the above if statements can be executed the
next interation

  fill(33);
  textSize(32);
  text("X: ", 500, 290);
  text(xP, 533, 290);
  text("Y: ", 650, 290);
  text(yP, 685, 290);
  text("Z: ", 800, 290);
  text(zP, 835, 290);
  textSize(26);
  text("Gripper", 650, 420);
  text("CLOSE", 510, 470);
  text("OPEN", 810, 470);
  textSize(18);

  if (positionsCounter >0 ) {
    text(positions[positionsCounter-1], 460, 630);
    text("Last saved position: No."+(positionsCounter-1), 460, 600);
  } else {
    text("Last saved position:", 460, 600);
    text("None", 460, 630);
  }
}

 // FORWARD KINEMATICS
void forwardKinematics() {
  float theta1F = theta1 * PI / 180;   // degrees to radians
  float theta2F = theta2 * PI / 180;
  xP = round(L1 * cos(theta1F) + L2 * cos(theta1F + theta2F));
  yP = round(L1 * sin(theta1F) + L2 * sin(theta1F + theta2F));
}

 // INVERSE KINEMATICS
void inverseKinematics(float x, float y) {
  theta2 = acos((sq(x) + sq(y) - sq(L1) - sq(L2)) / (2 * L1 * L2));
  if (x < 0 & y < 0) {
    theta2 = (-1) * theta2;
  }

  theta1 = atan(x / y) - atan((L2 * sin(theta2)) / (L1 + L2 * cos(theta2)));

  theta2 = (-1) * theta2 * 180 / PI;
  theta1 = theta1 * 180 / PI;

 // Angles adjustment depending in which quadrant the final tool coordinate x,y is
  if (x >= 0 & y >= 0) {       // 1st quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y > 0) {       // 2nd quadrant
    theta1 = 90 - theta1;
  }
  if (x < 0 & y < 0) {       // 3d quadrant
    theta1 = 270 - theta1;
    phi = 270 - theta1 - theta2;
    phi = (-1) * phi;
  }
  if (x > 0 & y < 0) {       // 4th quadrant
    theta1 = -90 - theta1;
  }
  if (x < 0 & y == 0) {
    theta1 = 270 + theta1;
  }

  // Calculate "phi" angle so gripper is parallel to the X axis
  phi = 90 + theta1 + theta2;
  phi = (-1) * phi;

  // Angle adjustment depending in which quadrant the final tool coordinate x,y is
  if (x < 0 & y < 0) {       // 3d quadrant
    phi = 270 - theta1 - theta2;
  }
  if (abs(phi) > 165) {
    phi = 180 + phi;
  }

  theta1=round(theta1);
  theta2=round(theta2);
  phi=round(phi);

  cp5.getController("j1Slider").setValue(theta1);
  cp5.getController("j2Slider").setValue(theta2);
  cp5.getController("j3Slider").setValue(phi);
  cp5.getController("zSlider").setValue(zP);
}

void controlEvent(ControlEvent theEvent) {

  if (theEvent.isController()) {
    println(theEvent.getController().getName());
  }
}

public void xTextfield(String theText) {
  //If we enter a value into the Textfield, read the value, convert to integer, set the
inverseKinematics mode active
  xP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP); // Use inverse kinematics to calculate the J1(theta1), J2(theta2), and
J3(phi) positions
  //activeIK = false;
  println("Test; theta1: "+theta1+" theta2: "+theta2);
}
```

```
public void yTextfield(String theText) {
  yP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP);
  //activeIK = false;
}
public void zTextfield(String theText) {
  zP=Integer.parseInt(theText);
  activeIK = true;
  inverseKinematics(xP, yP);
}

public void j1JogMinus() {
  int a = round(cp5.getController("j1Slider").getValue());
  a=a-j1JogValue;
  cp5.getController("j1Slider").setValue(a);
}
//J1 control
public void j1JogPlus() {
  int a = round(cp5.getController("j1Slider").getValue());
  a=a+j1JogValue;
  cp5.getController("j1Slider").setValue(a);
}
//J2 control
public void j2JogMinus() {
  int a = round(cp5.getController("j2Slider").getValue());
  a=a-j2JogValue;
  cp5.getController("j2Slider").setValue(a);
}
public void j2JogPlus() {
  int a = round(cp5.getController("j2Slider").getValue());
  a=a+j2JogValue;
  cp5.getController("j2Slider").setValue(a);
}
//J3 control
public void j3JogMinus() {
  int a = round(cp5.getController("j3Slider").getValue());
  a=a-j3JogValue;
  cp5.getController("j3Slider").setValue(a);
}
public void j3JogPlus() {
  int a = round(cp5.getController("j3Slider").getValue());
  a=a+j3JogValue;
  cp5.getController("j3Slider").setValue(a);
}
//J3 control
public void zJogMinus() {
  int a = round(cp5.getController("zSlider").getValue());
  a=a-zJogValue;
  cp5.getController("zSlider").setValue(a);
}
public void zJogPlus() {
  int a = round(cp5.getController("zSlider").getValue());
  a=a+zJogValue;
  ;
  cp5.getController("zSlider").setValue(a);
}

public void move() {

  myPort.write(data);
  println(data);
}

public void savePosition() {
  // Save the J1, J2, J3 and Z position in the array
  positions[positionsCounter]="J1="+str(round(cp5.getController("j1Slider").getValue()))
    +"; J2=" + str(round(cp5.getController("j2Slider").getValue()))
    +"; J3="+str(round(cp5.getController("j3Slider").getValue()))
    +"; Z="+str(round(cp5.getController("zSlider").getValue()));
  positionsCounter++;
  saveStatus = 1;
  updateData();
  myPort.write(data);
  saveStatus=0;
}

public void run() {

  if (runStatus == 0) {
    cp5.getController("run").setCaptionLabel("STOP");
    cp5.getController("run").setColorLabel(#e74c3c);

    runStatus = 1;
  } else if (runStatus == 1) {
    runStatus = 0;
    cp5.getController("run").setCaptionLabel("RUN PROGRAM");
    cp5.getController("run").setColorLabel(255);
  }
  updateData();
  myPort.write(data);
}
public void updateSA() {
  myPort.write(data);
}
public void clearSteps() {
  saveStatus = 2; // clear all steps / program
  updateData();
  myPort.write(data);
  println("Clear: "+data);
  positionsCounter=0;
  saveStatus = 0;
}
```

```
public void updateData() {
  data = str(saveStatus)
    +","+str(runStatus)
    +","+str(round(cp5.getController("j1Slider").getValue()))
    +","+str(round(cp5.getController("j2Slider").getValue()))
    +","+str(round(cp5.getController("j3Slider").getValue()))
    +","+str(round(cp5.getController("zSlider").getValue()))
    +","+str(gripperValue)
    +","+str(speedSlider)
    +","+str(accelerationSlider);
}
```

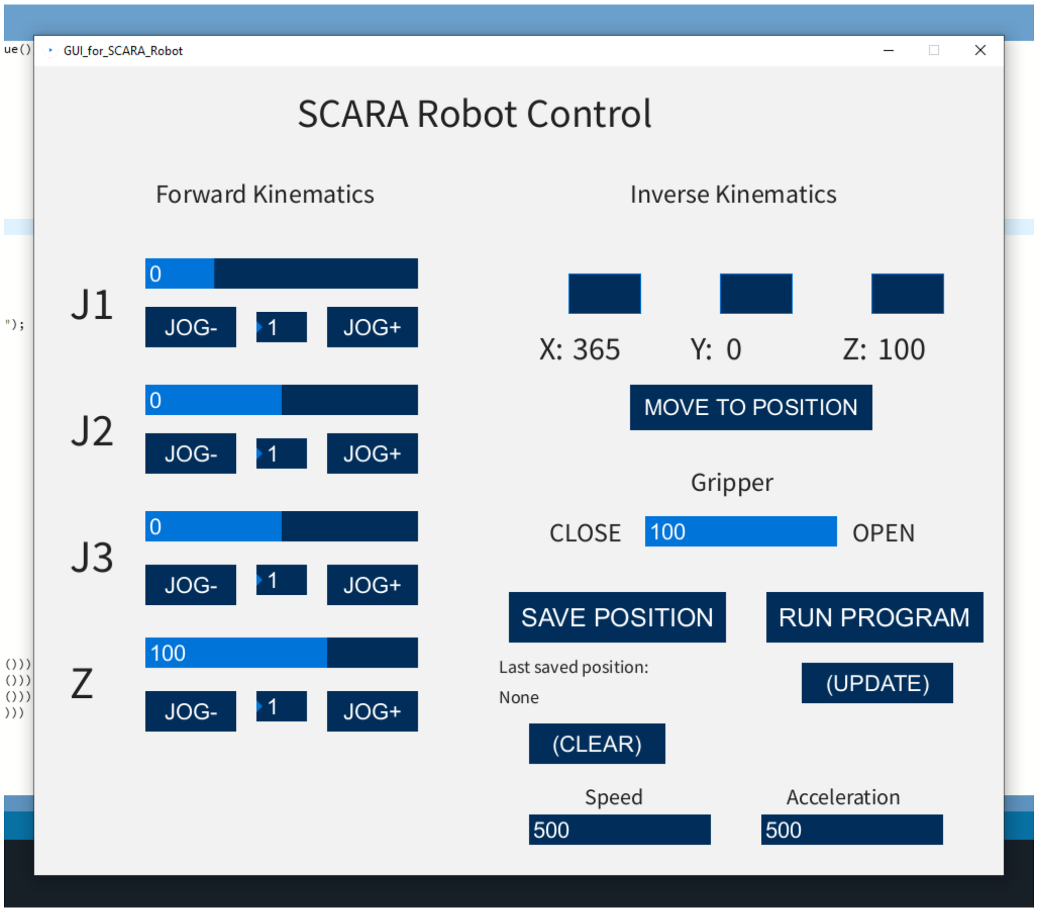


FIGURE 49

GUI for SCARA Robot

References

[1] *"SCARA Robots - Fanuc". www.fanuc.eu. Retrieved 2021-05-27.*

[2] ^ Jump up to:*a b* *"The Robot Hall of Fame - Powered by Carnegie Mellon University". www.robothalloffame.org. Retrieved 2021-05-27.*

[3] ^ *Wu, Guanglei; Shen, Huiping (2020-08-08). Parallel PnP Robots: Parametric Modeling, Performance Evaluation and Design Optimization. Springer Nature. ISBN 978-981-15-6671-4*

[4] Mike Wilson, in Implementation of Robot Systems, 2015

[Adept, n.d.]

Adept – SCARA robots. (n.d.). Retrieved from http://www.adept.com/products/robots/scara/cobra-i600/general.

[Ben-Gharbia et al., 2014]

K.M. Ben-Gharbia, A.A. Maciejewski, R.G. Roberts.

A kinematic analysis and evaluation of planar robots designed from optimally fault-tolerant Jacobins.

IEEE Transactions on Robotics, 30 (2014), pp. 516-524

http://dx.doi.org/10.1109/TRO.2013.2291615

[Bruzzone and Bozzini, 2011]

L. Bruzzone, G. Bozzini.

A statically balanced SCARA-like industrial manipulator with high energetic efficiency.

Meccanica, 46 (2011), pp. 771-784

http://dx.doi.org/10.1007/s11012-010-9336-6

[Gómez et al., 2014]

A. Gómez, P.D. Lafuente, C. Rebollar, M.A. Hernández, E.H. Olguín, H. Jiménez, …, J. Rodríguez.

Design and construction of a didactic 3-dof parallel links robot station with a 1-dof gripper.

Journal of Applied Research and Technology, 12 (2014), pp. 435-443

http://dx.doi.org/10.1016/S1665-6423(14)71624-4

[Jo and Cheol, 2001]

S. Jo, M. Cheol.

Design of a fuzzy-sliding mode controller for a SCARA robot to reduce chattering.

KSME International Journal, 15 (2001), pp. 339-350 http://dx.doi.org/10.1007/BF03185217

[López et al., 2013]

I. López, M. Castelán, F.J. Castro, M. Peña, R. Osorio.

Using object's contour, form and depth to embed recognition capability into industrial robots.

Journal of Applied Research and Technology, 11 (2013), pp. 5-17

http://dx.doi.org/10.1016/S1665-6423(13)71511-6

[Mathworks, n.d.]

Mathworks. (n.d.). Retrieved from http://www.mathworks.com/help/matlab/.

[Prajumkhaiy and Mitsantisuk, 2016]

N. Prajumkhaiy, C. Mitsantisuk.

Sensorless force estimation of SCARA robot system with friction compensation.

Procedia Computer Science, 86 (2016), pp. 120-123

http://dx.doi.org/10.1016/j.procs.2016.05.030

[Rossomando and Soria, 2016]

F.G. Rossomando, C.M. Soria.

Discrete–time sliding mode neuro-adaptive controller for SCARA robot arm.

Neural Computing and Applications, (2016), pp. 1-14

http://dx.doi.org/10.1007/s00521-016-2242-7

[Siciliano and Khatib, 2008]

B. Siciliano, O. Khatib.

Springer handbook of robotics.

1st ed., Springer, (2008),

[Siqueira et al., 2011]

A. Siqueira, M. Terra, M. Bergerman.

Robust control of robots.

Springer-Verlag, (2011), http://dx.doi.org/10.1007/978-0-85729-898-0

[Southern Plantaids Pvt.Ltd, n.d.]

Southern Plantaids Pvt.Ltd. (n.d.). Retrieved from http://www.southernplantaids.co.in/zasche-manipulators.html.

A. Nagchaudhuri et al.

Introduction of mechatronics concepts in a robotics course using an industrial SCARA robot equipped with a vision sensor

Mechatronics

(2002)

K.S. Hong et al.

A PC-based open robot control system: PC-ORC

Robotics and Computer Integrated Manufacturing

(2001)

M.J. Er et al.

Real time hybrid adaptive fuzzy control of a SCARA robot

Microprocessors and Microsystems

(2001)

S.S. Ge et al.

A robust distributed controller of a single-link SCARA/Cartezian smart materials robot

Mechatronics

(1999)

P. Bhatia et al.

An expert system-based design of SCARA robot

Expert Systems with Applications

(1998)

R.C. Dorf

Concise International Encyclopedia of Robotics—Applications and Automation

(1990)

S.Y. Nof

Handbook of Industrial Robotics

(1985)

R.I. Eugene

Mechanical Design of Robots

(1988)

A. Omodei et al.

Three methodologies for the calibration of industrial manipulators: experimental results on a SCARA robot

Journal of Robotic Systems

(2000)