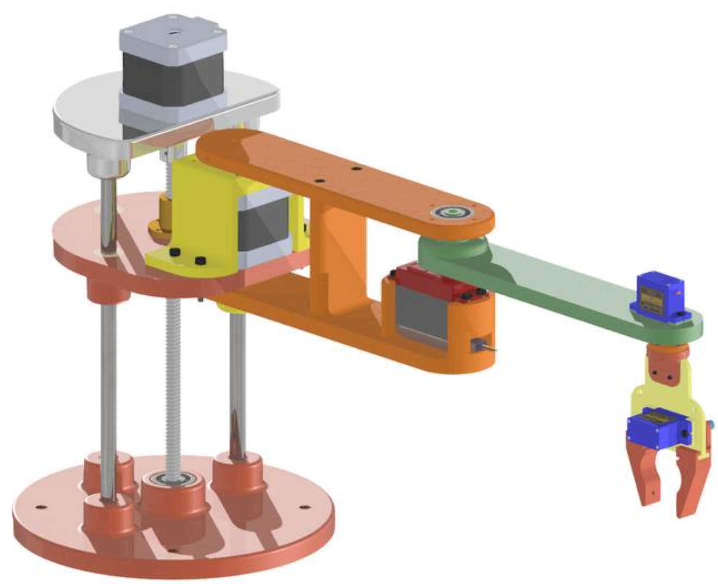
Electrical Engineering Department,

Fourth Year - Communications & Electronics.

PRR SCARA 5-DOF ARM Controlled by Mobile APP



Name	ID
احمد محمد احمد عبدالجواد	18010198
احمد محمد احمد ابوجبل	18010200
احمد العربي احمد علي	18010068
محمد طارق محمد عامر	18011505
آسر محمد عاطف زايد	18010299
علي محمد طعيمة حمص	18011084
احمد عاطف احمد اصيل	18010150
ضياء محمد الاباصيري	18010870
محمد محمود محمد الدملاوي	17015018
يمنى محمد رفعت عباس	18012098

https://github.com/youmna2023/PRR-SCARA-5-DOF-ARM

https://drive.google.com/drive/folders/16I3m4DXi3jt5YGDzKDWEBkbBe_-X9UjB?usp=sharing

➢ Contents

Introduction:

Nowadays, the objective of production at high speeds with low costs and low error rates in industrial production lines has gained great importance in terms of competitiveness. For this reason, companies often use different types of robots, such as Cartesian, SCARA, etc., in industrial applications. Cartesian systems are widely used in high-density warehouses and, generally, have both shuttle and aisle robots that generate the Cartesian structure. SCARA (Selective Compliance Assembly Robot Arm) manipulators take up less space than Cartesian systems, are easier to install, and can operate without the need for large areas. For this reason, processes such as packaging, sorting, alignment, planar welding, and assembly in the production lines are usually performed with SCARA-type manipulators. The first SCARA robot was developed in 1978 by Professor Hiroshi Makino at Yamanashi University in Japan. Afterwards, many types of SCARA robots have emerged to be used in the machine, automotive, and robot industries.

In literature studies, kinematic and dynamic modeling, simulation analysis, different control methods, and trajectory planning have been studied both theoretically and experimentally. Different decentralized and centralized (model-based) controllers have been tested with experimental studies of an industrial SCARA robot.

As a result, the performance of decentralized controllers was found to be sufficiently accurate for a large number of industrial applications. Accurate results of experimental studies depend on well-made mathematical modeling. In SCARA robots, which are generally used in industrial applications, it is very important to make both dynamic and kinematic calculations accurately in order to make the system work properly. While Das and Dulger developed a complete mathematical model with actuator dynamics and motion equations derived by using the Lagrangian mechanics, Alshamasin et al. investigated kinematic modeling and simulation of a SCARA robot by using solid dynamics by means of MATLAB/Simulink. Unlike other studies, Urrea and Kern implemented a simulation of a 5-Degree-Of-Freedom (DOF) SCARA manipulator using MATLAB/Simulink software. Their study has no physical application, although it bears similarity with the work we have done. This study enjoys some advantages over these types of works, which include only modeling and simulation. Kaleli et al. and Korayem et al. designed a program for simulating and animating robot kinematics and dynamics in LabView software. Similar to these works, there are various robot control, simulation, and calculation program studies in the literature. While some of them are just based on the analysis and simulation of one type of robot arms, some give results for robots in different types.

SCARA robots with RRP(Revolute-Revolute-Prismatic) or PRR (Prismatic-Revolute-Revolute) joint configurations are easy to provide linear movement in vertical directions. RRP and PRR types have some advantages and disadvantages. RRP-type SCARA manipulators are very common in light-duty applications that require precision and speed, which is difficult to achieve by human beings. While the prismatic joint motor is only lifting the objects in RRP type, it is lifting the whole robot structure with the objects in PRR type. Therefore, the prismatic joint motor of PRR type has higher torque than that of RRP type. Therefore, the PRR-type SCARA robot configuration is preferred in applications, where lifting heavy weights is a challenge. Since the base is fixed on one-point, powerful torque motors for lifting heavy loads linearly can be used easily.

In this study, a PRR-type (Prismatic-Revolute-Revolute) SCARA robot manipulator is designed. In addition, a gripper is placed on the last joint so that the objects can be picked and placed at the desired locations.

Used Components:

- ## Electrical Components:

    1. Motors:
        - DC
        - Servo
        - Stepper
    2. Wires
    3. Bread Board
    4. L298 DC controller
    5. 100 uF Capacitor
    6. Stepper driver
    7. 12V Supply

- ## Mechanical Components:

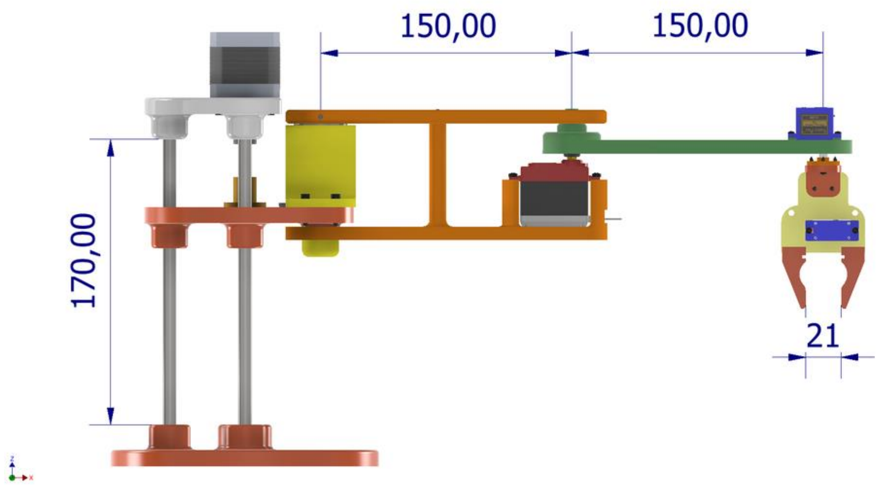
    1. 3D Printing
    2. Bearing
    3. Rods
    4. Coupler
    5. Lead screw

System and methods:

Mechanical design:

https://grabcad.com/library/simple-robot-scara-for-3d-printing-1

https://drive.google.com/file/d/16L42If7MvQ1B7LhJYSfk5EToQm7eYmhB/view?usp=sharing



PRR type SCARA manipulator is designed in SolidWorks 2020. The linear motion in the prismatic joint is achieved with the lead screw and stepper motor arrangement. The manipulator can traverse 450 mm on the z-axis. Likewise, lengths of 150 mm and 150 mm are taken for the first and second arm, respectively. Timing gear of 1.25 module and 20 teeth is employed to transfer motion from the stepper motor shaft. Likewise, a pulley of 1.25 module and 63 teeth modeled onto the arm ends is employed for the transmission of required motion. Arms bearing teethed wheels are fabricated through additive manufacturing.

Let us assume that it is possible to introduce four main parameters typically known as DH parameters to describe the individual link of the manipulator. A reference frame for each link is defined to obtain DH parameters which are shown in Fig. 1.

The notations used in the figure are as follows:

• aj is the common normal distance measured from the axis of j to the axis of j + 1

• α is the angle by which the axis of j must be twisted to align it with the axis of j + 1 while looking along aj

• dj is the distance between the two normal aj − 1 and aj measured along the joint axis from aj − 1 to aj

• θj is the joint angle from aj − 1 to aj in the plane normal to the joint axis

DH parameters computed assuming the direction of x-axis for the first joint being coincident with the direction of first link is illustrated in Table 1.

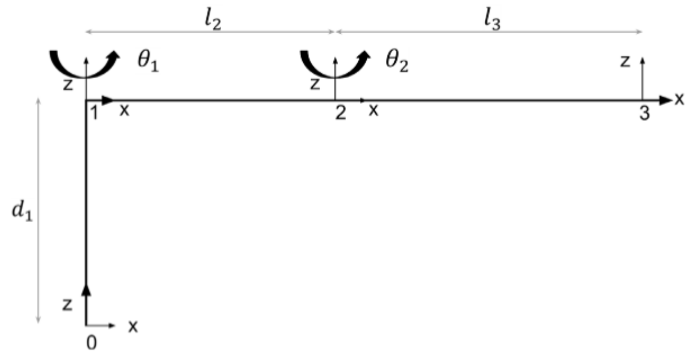
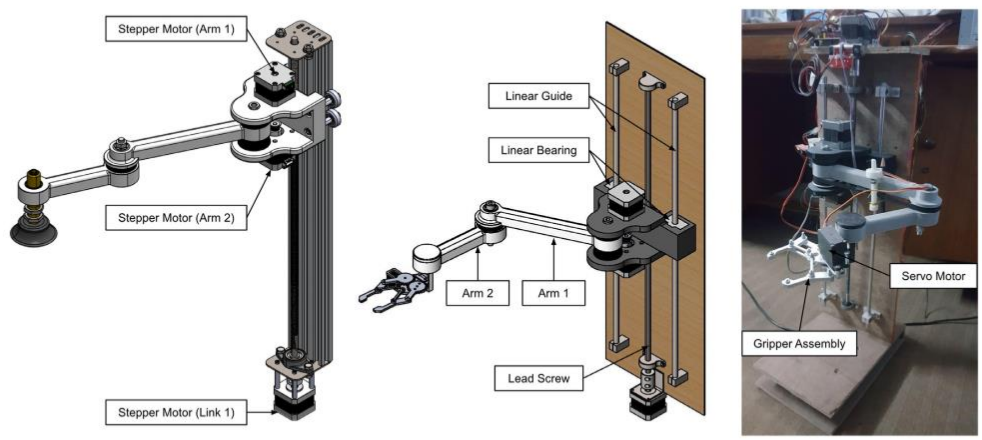
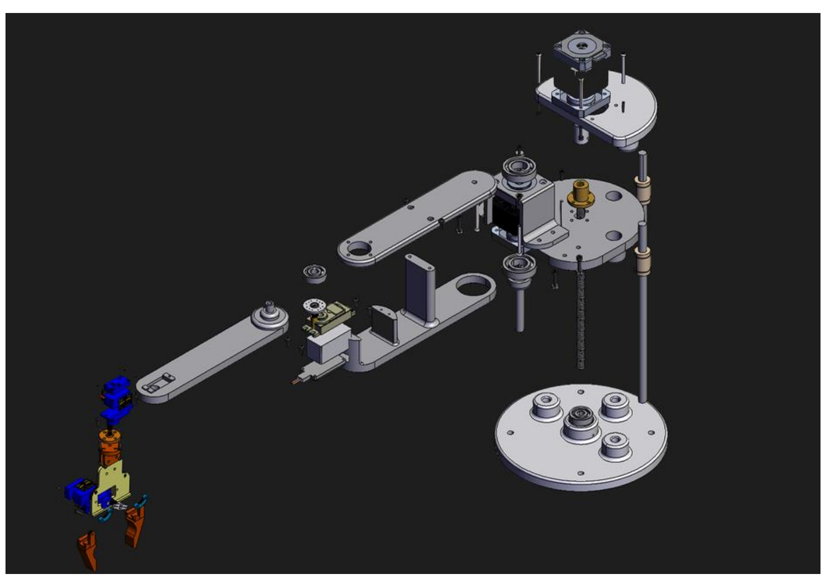


Fig. 1. Joint axes in PRR type SCARA.

Table 1
DH parameters for 3 DOF SCARA.

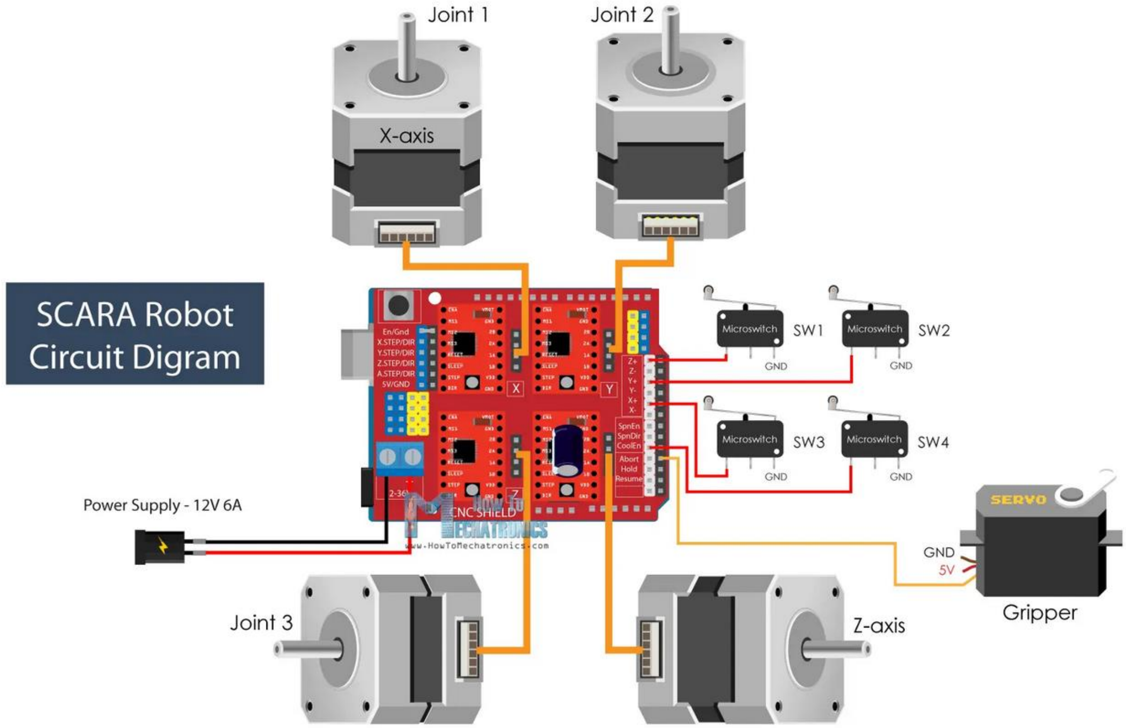
Joint	α_j	a_j	θ_j	d_j
1	0	0	0	d_1
2	0	l_2	θ_1	0
3	0	l_3	θ_2	0

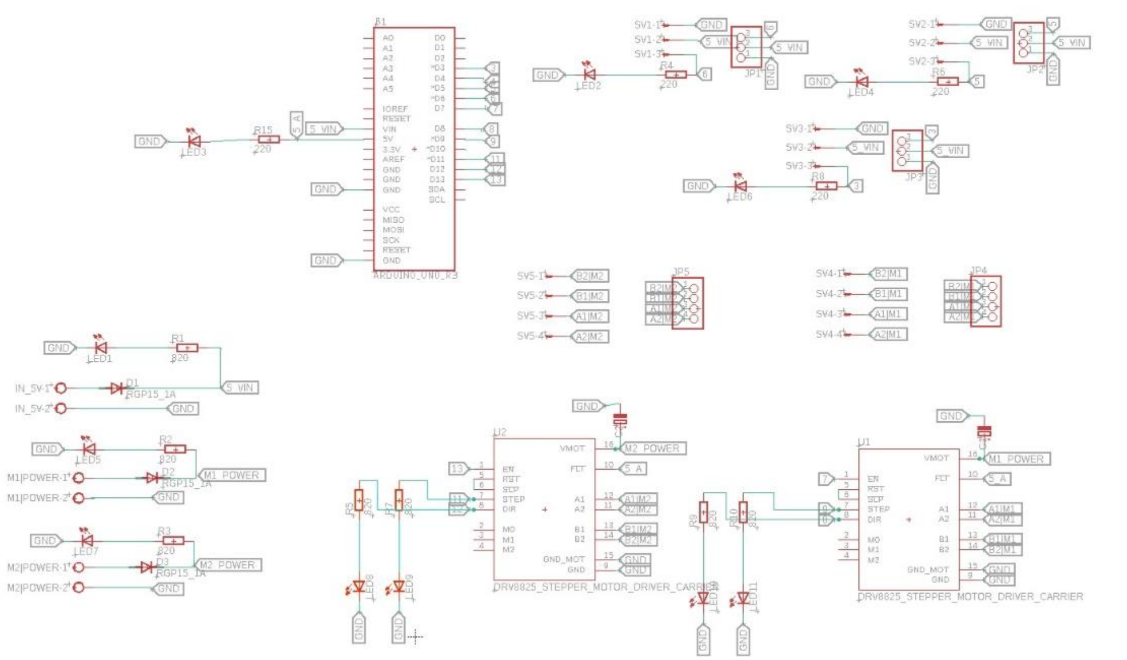


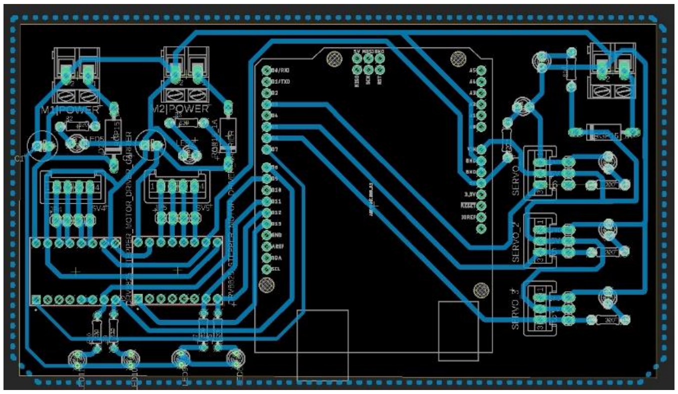


Control System:

The heart of the control system of the SCARA manipulator is an Arduino Uno microcontroller board.





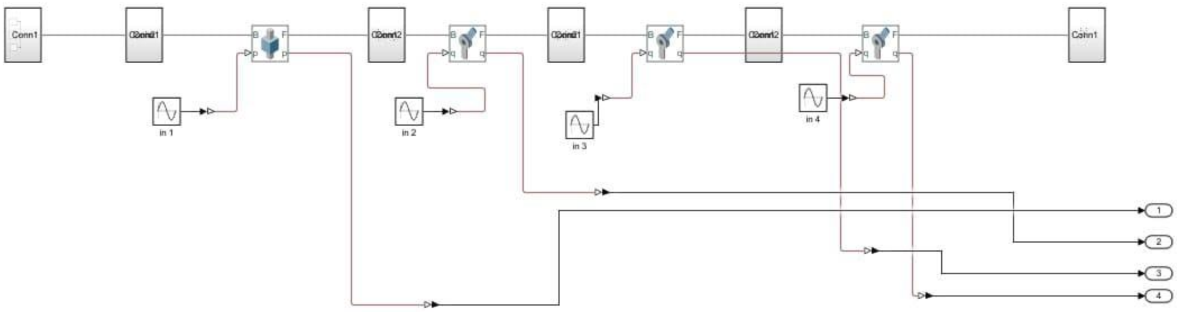


Vision system:

The vision and path planning system are integrated to microcontroller through MATLAB Interface running in a PC. The MATLAB Support Package for Arduino is used to control the system which runs in MATLAB environment. From the detection coordinates obtained, inverse kinematics is used to calculate required joint angles and parameters. Inverse kinematics, path-planning and the desired trajectory are calculated through MATLAB code which is then transmitted to the microcontroller. Though all the calculations take place in MATLAB, only desired angular data is transmitted to the microcontroller to limit the communication between MATLAB and microcontroller. Moreover, as the SCARA arm resets itself to home position for each new operation recalibrating the angular system is required only after long periods of operation.

Forward Kinematics:

https://drive.google.com/file/d/1IwmE621urmvCXtMaQcz60qNTPPb6Xanc/view?usp=sharing



Inverse Kinematics:

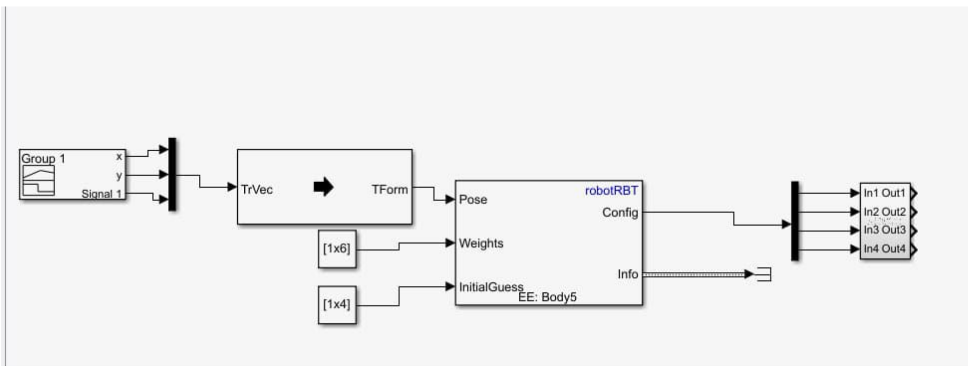
https://drive.google.com/file/d/1s7kelYkmIOwUH8YQ46LmTLwm-9MGNun8/view?usp=sharing

```
clc

clear all

%Ts=0.001;

[robotRBT,robotData] =
importrobot('test.slx');
```

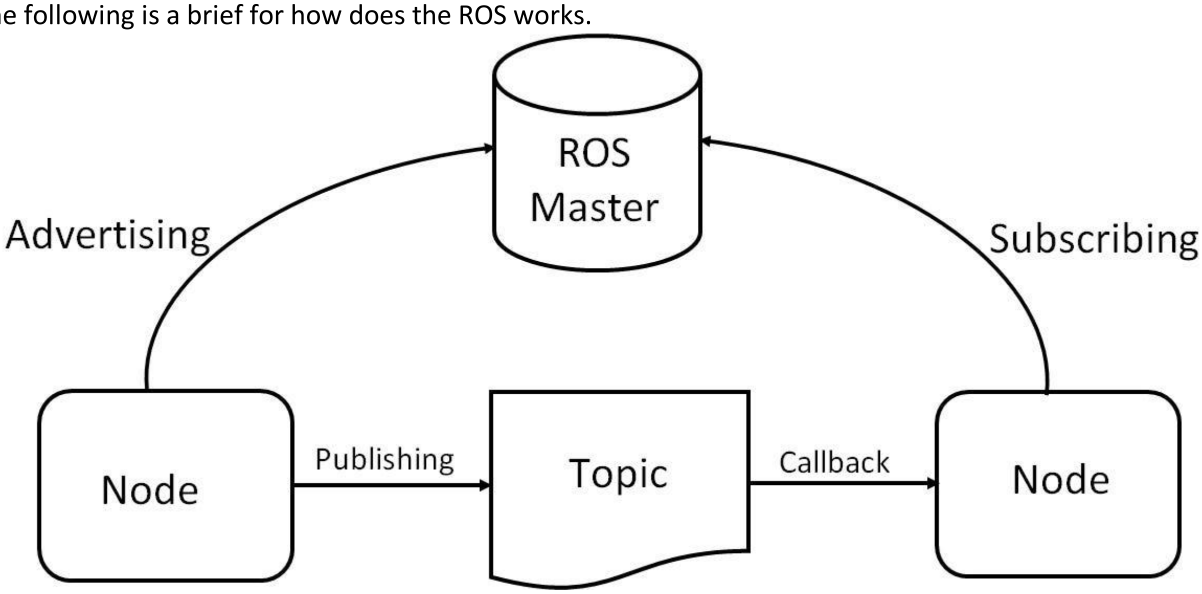


Codes

https://drive.google.com/file/d/16I_DFZB8_deuWOOWuWYz2Us6BkeyR9e5/view?usp=sharing

Intro

For the software system, we used ROS as our main Framework and communication with the Arm. So, the following is a brief for how does the ROS works.



ROS, or Robot Operating System, is an open-source framework that is widely used in robotics applications. It provides a set of tools, libraries, and conventions to help developers create complex software systems for robots.

At its core, ROS is a message-passing framework that enables communication between different software components of a robot system. ROS uses a publish-subscribe model, where nodes (individual software components) can publish messages to a topic, and other nodes can subscribe to that topic to receive those messages.

In our specific software system, we used ROS as the main framework for communication with the arm. This involved creating ROS nodes that communicated with the arm's hardware drivers, as well as other nodes that processed sensor data and executed high-level control commands.

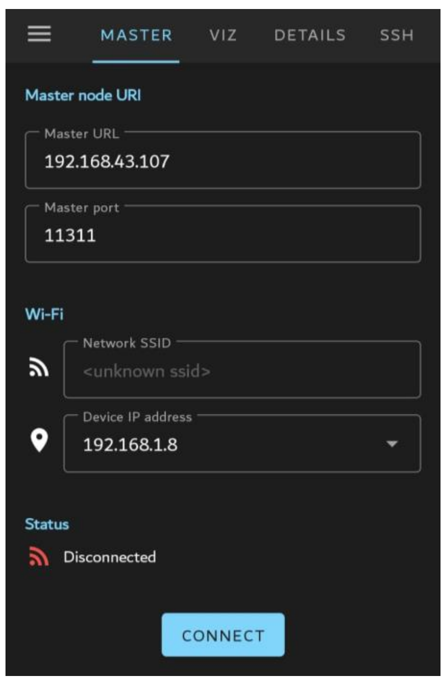
For ROS to be functional we need to set a single Master to handle the topic publishing and subscribing, also specifying the ROS master with network IP allows it to handle the nodes and topic across multiple devices on the same network.

Main component

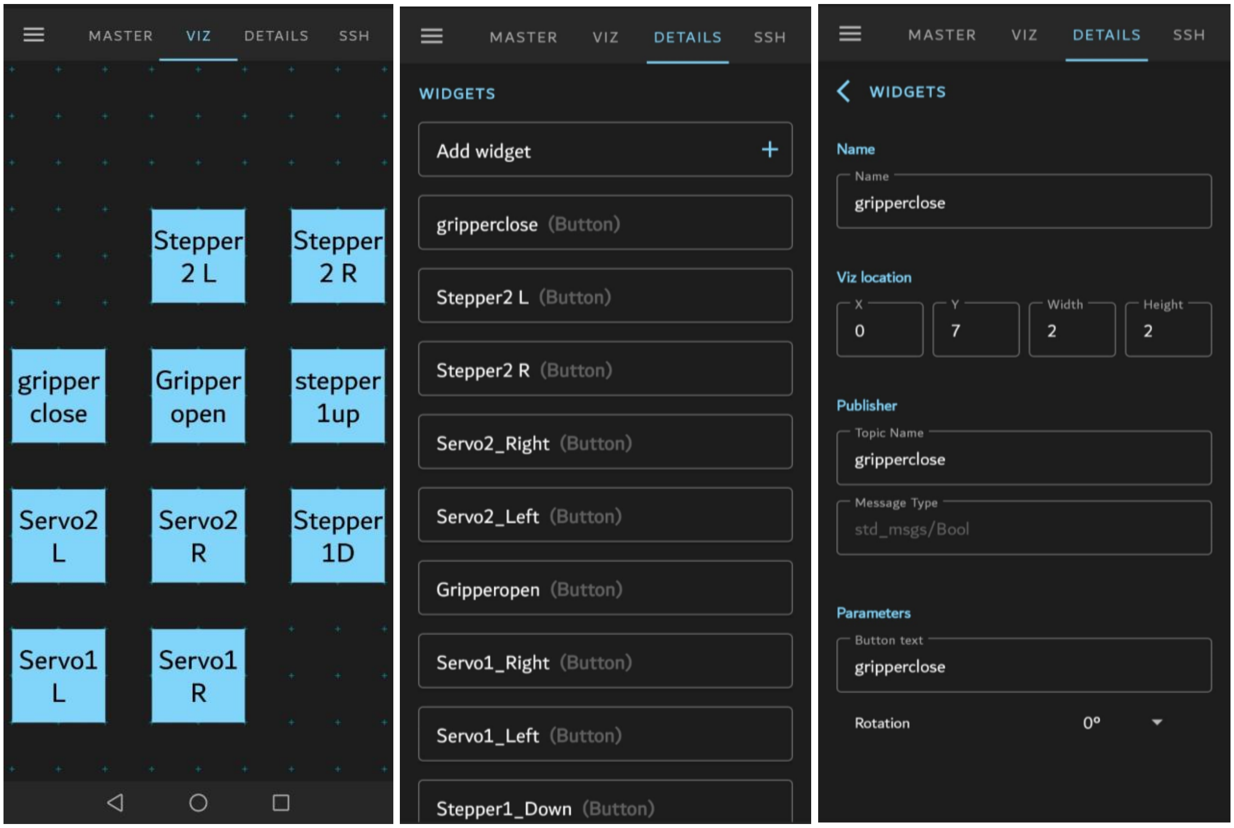
Mobile app

We used ROS-Mobile app to send commands through a mobile application.

We Set the network configuration on the app so we can connect to the ROS master.



We also set a custom GUI which in our case consists of buttons only. Each button is set to publish a Bool value on a specific topic in the network. This Bool Value simply says which motor should move and in which direction.



Python node

The Python node simply subscribes to the topic from the Mobile app.

```
    rospy.Subscriber("stepperup",Bool,stepper1UP_cb)
```

It publishes to another topic which contains a single int value that will decide which action should be taken.

```
    pub=rospy.Publisher("android_topic" , Int8 , queue_size=1)
```

Each subscriber has a call back function which sets the value that will be published.

```
def stepper1UP_cb (msg):
    if msg.data == 1:
        stepper1UP=1
    else:
        stepper1UP=0
    pub.publish (stepper1UP)
    rate.sleep()
    rospy.loginfo(stepper1UP)
```

Arduino node

The Arduino node simply subscribes to the topic that was published by the python node.

```
ros::Subscriber<std_msgs::Int8> android_sub("/android_topic", &androidCallback);
```

and same as the python node it has a callback function that changes a variable which will determine the desired action.

```
void androidCallback(const std_msgs::Int8& msg)
{
  nh.loginfo("Callback");
  motion = msg.data;
}
```

In the loop function a if-else structure was made to move each motor depending on the desired motion and the type of the motor.

```
void loop() {
  //nh.loginfo(motion);
  if(motion==3)
  {
    digitalWrite(STEPPER2_DIR_PIN,0);
    digitalWrite(STEPPER2_STEP_PIN,1);
    delayMicroseconds(1000);
    digitalWrite(STEPPER2_STEP_PIN,LOW);
    delayMicroseconds(1000);
  }
  else if(motion==4)
  {
    digitalWrite(STEPPER2_DIR_PIN,1);
    digitalWrite(STEPPER2_STEP_PIN,1);
    delayMicroseconds(1000);
    digitalWrite(STEPPER2_STEP_PIN,LOW);
    delayMicroseconds(1000);
  }
  else
  {
    digitalWrite(STEPPER2_DIR_PIN,0);
    digitalWrite(STEPPER2_STEP_PIN,0);
    delayMicroseconds(1000);
    digitalWrite(STEPPER2_STEP_PIN,LOW);
    delayMicroseconds(1000);
    if(motion==1)
    {
      digitalWrite(DC11, HIGH);
      digitalWrite(DC12, LOW);
    }
    else if(motion==2)
```

```
    {
      digitalWrite(DC11, LOW);
      digitalWrite(DC12, HIGH);
    }
    else if(motion==5)
    {
      moveServo1Right();
    }
    else if(motion==6)
    {
      moveServo1Left();
    }
    else if(motion==7)
    {
      moveServo2Right();
    }
    else if(motion==8)
    {
      moveServo2Left();
    }
    else if(motion==9)
    {
      servo3.write(0);
    }
    else if(motion==10){
      servo3.write(90);
    }
    else
    {
      digitalWrite(DC11, LOW);
      digitalWrite(DC12, LOW);
      nh.loginfo("Waiting");
    }
  }
  nh.spinOnce();
  delay(10);
}
```

Steps and summary

1. Set up the network.
2. Run ROS master.
3. Connect the Mobile app to the ROS master.
4. Run the Python node.
5. Run the Arduino node.
6. Start sending orders, we are done.

Budget

DC	**200**
Servo-Small	**75 * 2 = 150**
Servo	**185**
Stepper	**150**
L298	**80**
Stepper driver	**80**
12V Supply	**50**
3D Printing	**600**
Bearing	**75**
Rods	**30 * 3 = 90**
Coupler	**60**
Lead screw	**50**

Total= 1770

References

1. Robot Hall of Fame, \Inductees-SCARA", Carnegie Mellon University (2006). http://www.robothalloffame.org/inductees/06inductees/scara.html
2. Visioli, A. and Legnani, G. \On the trajectory tracking control of industrial SCARA robot manipulators", IEEE Transactions on Industrial Electronics, 49(1), pp. 224{232 (2002).
3. Das, M.T. and Dulger, L.C. \Mathematical modelling, simulation and experimental verification of a SCARA robot", Simulation Modelling Practice and Theory, 13(3), pp. 257{271 (2005).
4. Alshamasin, M.S., Ionescu, F., and Al-Kasasbeh, R.T. \Kinematic modelling and simulation of a scara robot by using solid dynamics and verification by MATLAB/Simulink", European Journal of Scientific Re-search, 37(3), pp. 388{405 (2009).
5. Urrea, C. and Kern, J. \Modelling, simulation and control of a redundant SCARA-type manipulator robot", International Journal of Advanced Robotic Systems, 9(2), p. 58 (2012).
6. Kaleli, A., Dumlu, A., Corapsz, M.F., and Erenturk,K. \Detailed analysis of SCARA type serial manipulator on a moving base with LabVIEW", International Journal of Advanced Robotic Systems, 10(4), p. 189 (2013).
7. Korayem, M.H., Yousefzadeh, M., and Manteghi, S.\Tracking control and vibration reduction of flexible cable-suspended parallel robots using a robust input shaper", Scientia Iranica B, 25(1), pp. 230{252 (2018).
8. Kozkurt, C. and Soyaslan, M. \Software development for kinematic analysis of scara robot arm with Euler wrist", 6th International Advanced Technologies Symposium (IATS'11), Elazg, Turkey, pp. 27{32 (2011).
9. Kucuk, S. and Bingul, Z. \An o -line robot simulation toolbox", Computer Applications in Engineering Education, 18(1), pp. 41{52 (2009).

10. Adar, N.G. and Kozan, R. \Comparison between real time PID and 2-DOF PID controller for 6-DOF robot arm", Acta Phys. Pol. A, 130(1), pp. 269{271 (2016).
11. Adar, N.G., Tiryaki, A.E., and Kozan, R. \Real time visual servoing of a 6-DOF robotic arm using Fuzzy- PID controller", Acta Phys. Pol. A, 128(2B), pp. 348{351 (2015).
12. Saygin, A. and Rashid, A.M. \Position control of a turret using LabVIEW", Acta Phys. Pol. A, 132(3-II),pp. 970{973 (2017).
13. Karayel, D. and Yegin, V. \Design and prototype manufacturing of a torque measurement system", Acta Phys. Pol. A, 130(1), pp. 272{275 (2016).
14. Ibrahim, B.S.K.K. and Zargoun, A.M.A. \Modelling and control of SCARA manipulator", Procedia Computer Science, 42, pp. 106{113 (2014).
15. Urrea, C., Cortes, J., and Pascal, J. \Design, construction and control of a SCARA manipulator with 6 degrees of freedom", Journal of Applied Research and Technology, 14(6), pp. 396{404 (2016).
16. Denavit, J. and Hartenberg, R.S. \A kinematic notation for lower-pair mechanisms based on matrices", ASME J. Appl. Mechan., 77(2), pp. 215{221 (1955).
17. Bingul, Z. and Kucuk, S. _Ileri kinematik, terskinematik", In Robot Teknigi I, pp. 104{200, Birsen Yaynevi, Turkey (2005).
18. TB6600 Stepper Motor Driver (2017). https://www.dfrobot.com/product-1547.html
19. Dynamixel-All in one actuator, Robotis Inc (2014). http://www.robotis.us/dynamixel/