WOLKITE UNIVERSITY

College of Engineering and Technology

Department of Electrical and Computer Engineering

Industrial Control Stream

Design and Implementation of Pick and Place Robot Arm Using ROS on Raspberry Pi

Prepared by:

Amanuel Ararso [ENGR/114/07]

Tadese Aberaham [ENGR/1033/07]

Eyerusalem Dirshaye [ENGR/401/07]

Advisor: Mr. Amare Ambaw

A Final-year Project report submitted for the partial fulfillment of the requirement for Bachelor of Science degree in Electrical and computer Engineering

June, 2019

Wolkite, Ethiopia

Declaration

We here by certify that the work titled by **Design and implementation of Pick and Place robot arm using ROS on Raspberry Pi** which submitted to Wolkite University Department of Electrical and Computer Engineering is an authentic record of the work done by Amanuel Ararso, Tadese Abereham and Eyerusalem Dirshaye under the supervision of Mr. Amare Ambaw. All the sources of the materials used by our work are properly referred.

Submitted by:

1. Amanuel Ararso [ENGR/114/07]
2. Tadese Aberaham [ENGR/1033/07]
3. Eyerusalem Dirshaye [ENGR/401/07]

Advisor: Mr. Amare Ambaw

Authors Name	Signature	Date
1. Amanuel Ararso [ENGR/114/07]	_____	_____
2. Tadese Abereham [ENGR/1033/07]	_____	_____
3. Eyerusalem Dirshaye [ENGR/401/07]	_____	_____

It is approved that this final-year project report has been written in compliance with the formatting rules laid down by the department of the university.

Examining Committe Members

	Name	**Signature**	**Date**
1. Department Head	***Habtamu Eshete***	_____	_____
2. Examiner 1	_____	_____	_____
3. Examiner 2	_____	_____	_____

Abstract

Pick and Place Robot arm is used in industry, mostly in manufacturing areas, product store and packaging areas. They reduce a remarkable time and effort. In Fincha Sugar Factory condensate water is preferred than the treated water for production of steam. Condensate water is a bled steam from evaporators, but as it passes many processes and transport to the boiler, sugar (glucose) content could be added at some point. This sugar containing condensate water then causes the boiler to burst and damage human life and/or materials. Currently, it is manually checked (in chemistry laboratory) every 30 minutes. This also couldn't save the boilers, and so two of four boilers currently are out of production. But, if this manual examination of the water is replaced by automatic examination, the boiler could be durable and the factory could increase production.

Although building the whole system could take a longer time, in this project the picking and placing the test cylinder part is done. The picking and placing task is designed to be done using two Degree of Freedom robot arm. The control of the arm is done by Robot Operating System (ROS) installing it on Raspberry Pi. Eight nodes are created, six nodes for each procedure for the execution, one node for control and one node for inverse kinematic solving server. Two topics are also created for communication of the current procedure and the goal procedure (stage). Arduino is used to actuate the servo motors of the arm which communicates through serial with Raspberry PI.

Acknowledgement

We would like to thanks to God first and foremost. Secondly, we would like to experess our deep gratitiude to our advisor Mr. Amare Ambaw for his supportive and corrective advise throughout the project time. We also have a great appreciation for Electrical and Computer Engineering department for the materials they made available to use.

Last but not least, we would like to thank our friends at the ECE association for their helpful support.

Table of Contents

List of Figures

Acronomy

ARM ……. Advanced RISC Machines

DH ……Denavit-Hartenberg

DOF ……Degree of Freedom

EE …….. End-Effector

GPIO……. General Purpose Inputs Outputs

HDMI …….High Definition Multimedia Interface

HTM ……. Homogenous Transformation Matrix method

IDE ……. Integrated Development Environment

IK……..… Inverse Kinematics

IP ……Internet Protocol

MIT…… Massachusetts Institute of Technology

OS ……. Operating System

OSRF ……. Open Source Robotics Foundation

RAM ……. Random Memory Access

RISC ……. Reduced Instruction Set Computer

ROS ……. Robot Operating System

RTOS …….Real-Time Operating System

RVIZ ……. Robot Visualizer

SCARA …….Selective Compliance Assembly Robot Arm

SoC …….System on a chip

URDF……Universal Robot Description Format

USB …….Universal Serial Buss

XML …….Extensible Markup Language

Chapter 1 Introduction

1.1. Background of the Project

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. The links of such a manipulator are connected by joints allowing either rotational motion or translational (linear) displacement. [1] The links of the manipulator form a kinematic chain and terminus of the kinematic chain of the manipulator is called the end effector and it is analogous to the human hand. Kinematics is the way of solving joint angles to move the end effector to a known position or using known joint angles, moving the end effector to a position i.e. inverse and forward kinematics respectively. In the decade of 2010, the availability of low- cost robotic arms increased substantially. Such robotic arms are mostly marketed as hobby or educational device. [1]

Robot arms are typically classified in terms of the number of degrees of freedom. Usually, the number of degrees of freedom is equal to the number of joints that move the links of the robot arm. Degrees of freedom may allow to change the configuration of some link on the arm (e.g., elbow up/down), while keeping the robot hand in the same pose. The designed robot arm for this project is of two degrees of freedom for pick and place application.

Pick and place robots are mostly industrial. It can be used in packaging work at the terminal of the production. These types of robots mostly have 2 Degrees of Freedom. For instance, Selective Compliance Assembly Robot Arm (SCARA) robot is a pick and place application robot which is utilized for high-speed pick and place manipulation. [2]

The software framework used in this project is Robot Operating System (ROS). It is robotics middleware, i.e. collection of software frameworks for robot software development. It provides services designed for a heterogeneous computer cluster such as, low-level device control, implementation of commonly used functionality like kinematics or sensors, message-passing between processes, and package management, although ROS is not an operating system, but it acts like it. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may subscribe to or publish sensor data, control, state, planning, actuator, and other messages. Despite the importance of reactivity and low latency in robot control, ROS itself is not a Real-Time OS (RTOS). It is possible, however, to integrate ROS with real-time code [2], as used in this project. For the purpose of this project, ROS is used because of it works on divide and

conquer principle, in logical manner, than using one long bunch of codes to control a robot. Adding other processes is fast and easy. Processes mean that adding sensors like camera, force sensors, adding third degree of freedom or changing the kinematics, does not require to re-code or re-design, it's all about adding another node and connect them

Our project, Design and Implementation of Pick and Place Robot Arm using ROS on Raspberry Pi, is done to be used in Fincha Sugar Factory condensate water treatment. In the factory, boiler produces steam using bagasse as a fuel. The raw material for this production is water. It could use two types of water, treated water from water treatment tank and condensate water from evaporators. If available, condensate water is preferred than the treated water. There are 4 condensate water tanks in the factory to supply water for the boilers.

This robotic arm is supposed to do the post examination work. Picking up the test cylinder and preparing it for next examination. For this purpose, only 2 Degrees of Freedom is enough to pick and move it to water outlet and placing it back. The first joint (base joint) of the arm, changes the orientation of the whole arm – like to moving in a circle. And the second joint (elbow) which moves the second link moves the link up and down.

1.2. Statement of the Problem

In Fincha Sugar Factory, condensate water from evaporators is used in boilers to produce steam. This condensate water is preferred than that of treated water, because of the parameters like hardness, PH and water temperature will be in better condition. This condensate water is normally the bled steam from the evaporators – the steam which is cooled back to water. In the process of the production, transportation and storage of this cooled steam (condensate water), sugar may be added at any level. But if this water with sugar content gets in to boilers, the boiler tubes definitely burst and cause fire, and since bagasse is all over the place around the plant, this fire and burst could endanger workers' life and destroy many million-birrs worth equipment.

This condensate water examination is currently done by quality control operators in every 30 minutes each hour and totally 16 times in one shift of eight hours. The examination room is far from condensate water tanks so sampler has to sample from every condensate tank again in every 30 minutes. But if this examination could be done frequently even in less than 5 minutes, the tube burst may not happen for a long time. Constructing the whole system to replace the current system takes even a year of development but it can be developed separating it into parts. Picking, washing and placing the cylinder is one of the sub-systems.

1.3. Objective

1.3.1. General Objective

In this project the main objective is building a 2 Degree of Freedom robotic arm that can pick and place a test cylinder and runs on Raspberry pi using ROS.

1.3.2. Specific Objectives

- Writing ROS nodes and services and connecting them
- Finding the forward kinematics and solving the inverse kinematics
- Integrating the factory's data in to ROS
- Integrating ROS on Raspberry Pi and configuring the workspace
- Serial communication of Raspberry Pi and Arduino
- Hacking U-arm to be 2 DOF robot arm

1.4. Scope and Limitation of the Project

1.4.1. Scope of the project

This project addresses only on the robotic arm which is to be used in post examination for single examination. The whole automated process of the examination is not the concern of this project. Because, the whole system takes longer time to fully design and implement. The robot arm is of 2 Degrees of Freedom only and the positions of the cylinder and water position are fixed and so that no image processing is used in this project to identify the objects size or position.

As this project is a prototype, only at the start the user commands the robot to start at the command line once and the robot stops when it finishes one cycle. The robot takes no environment sensors as it will be operating in clear environment with no obstacles – like industrial robots surrounded by glass. So that no motion planning is used in this project.

1.4.2. Limitation of the project

The motor used for joint angles is a servo motor and servos normally does not change the angle slowly as to not disturb the solution in the cylinder. Because of this, the speed of the servo is not managed in this project.

1.5. Motivation

In the factory, there was a boiler burst in 2005 E.C which killed 5 peoples and injured many. This is caused by careless examination of the condensate water and failure of exactness at the time of examination, since the examination is done manually. The factory has four boiler plants that can generate steam at the same time, but currently, only two of them are working due to the same problem. From the generated steam, electric power plant that can generate at 11kV is built but not working due to lack of steam. But if, the boiler probability to burst is decreased by careful examination, the factory can even sell the electricity and increase sugar production

1.6. Significance of the Project

This project is a part of a system that solves the problem of examination of condensate water in Fincha Sugar Factory and condensate water examination is very important for boiler durability and stable operation. If the current examination procedure is replaced by this proposed project, the examination could happen a lot faster, it could be stable day and night and reliable.

1.7. Outline of the Document

Chapter 2 describes the literatures reviewed while working on the project

Chapter 3 explains the methodology and hardware design with components description

Chapter 4 explains the software design and software components description

Chapter 5 deals with the results obtained and discussion

Chapter 6 lays out the conclusion concluded and recommendations for future works

Chapter 2 Literature Review

This section explains the literature studied that is related to the project task. It is from several sources such as books, websites and journals

D. Chikurtev et.al had proposed control of robotic arm manipulator using ROS [3]. This project had been done for simulation and debug purposes using ROS. Robot Visualizer (RVIZ) had been used as simulation environment. It is done as simple as possible by using kinematic solver called MoveIt. MoveIt is famous inverse kinematic solve which also can be used for motion planner. In this project, the robot structure is constructed using the Universal Robot Description Format (URDF) and then fed to the robot visualizer, RVIZ. And then MoveIt had been configured for the robot structure and it solved the inverse kinematics the robot had manipulated.

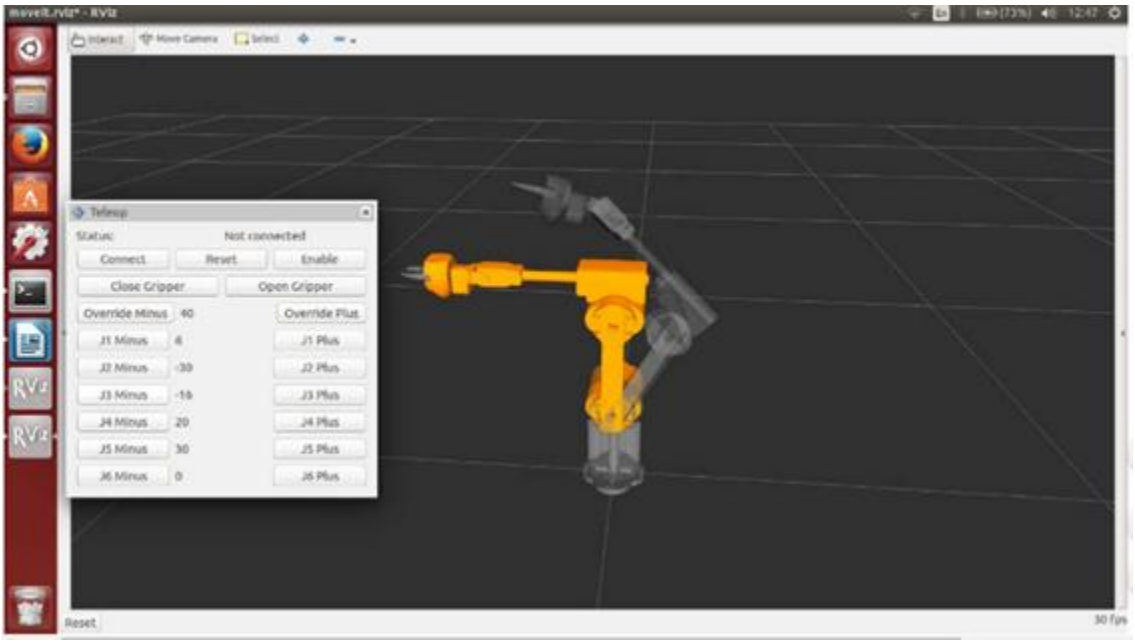


Figure 1: RVIZ simulation, from the paper

There are only the MoveIt and the simulator nodes and services are already doing the job rather than the project men. They only integrated the simulator with the MoveIt library for kinematics. However in our project, the forward and inverse kinematics is solved step by step, nodes and services which interact together are written as per the application. The hardware implementation also is done depending on the solved kinematics.

Ashly Baby et.al, [4] had proposed implementation of pick and place robotic arm using Arduino. It is designed to pick and place an object from source to destination safely. The soft catching gripper is used in the arm that helps not to apply any extra pressure on the objects. The robot had been designed to be controlled using android based smart phones through Bluetooth. Based on the commands given by the user the robot moves accordingly. At the receiver end there are four motors interfaced with the micro controller. Two for the vehicle movement and the remaining two are for arm and gripper movement. The application had been developed to send character commands like 'f', 'b', for forward and backward movement respectively and 'u' and 'd' for up and down movement of the arm

The proposed arm is 1 degree of freedom (DOF) and a gripper as the end-effector. The gripper had got a force sensor so that it does not harm the objects. Arduino takes the commands and reads the force sensor and powers the motors. The limitation of this proposed system is that since the DC motor are used there may not precisely go to the object and grip the object at the optimum place of grip.

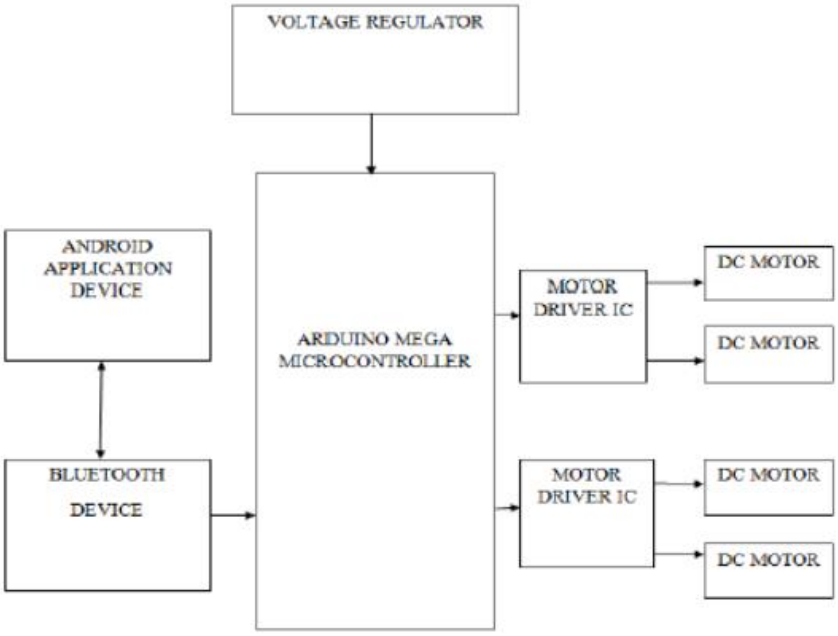


Figure 2:System block diagram, from the paper

Priyambada Mishra et.al, [5] had proposed a robot arm which is controlled by potentiometer using Arduino controller. The proposed system is for pick and place purpose and four servo

motors had been used. The mechanism of the robot structure was as of an articulated robot with three degrees of freedom and a gripper. The control mechanism depends on potentiometer control. Four potentiometers are used to control the four servo motor. They all are connected through analog input/output port and the value read from them, i.e. value from 0 to 1023 is mapped to servo angle which varies between 0 and 180. The system block diagram was as shown below.

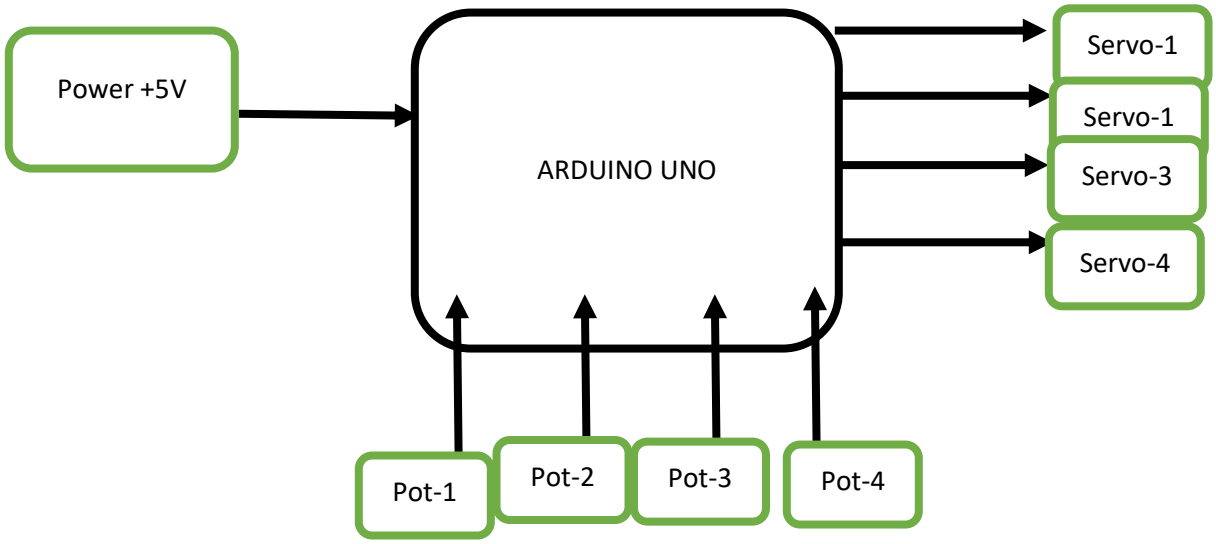


Figure 3: System block diagram, from the paper

This project is fully manual. There is a less computation time but it is a problem that it can not be used in industrial purposes. For full function, the human operator has to be operating the device. In software wise, map function of Arduino program is used. This has its own effect in precision, because its output is always an integer. In this thesis project but, the system is automatic and so that no human intervention is needed, unless only to start the arm.

In 1961, Unimate was the first industrial robot developed and it was a simple robotic arm implemented for automation of die-casting in the factory [6]. In another approach, Abhinav Sinha et.al have proposed a mathematical model for the control method of mobile pick and place industrial robots using inverted pendulum concept and have proven it mathematically. But no real-time implementation is carried out to prove its efficiency [7]. The entire

algorithm for the system is had been written in the same place so that, the system lags from real time clock in considerable amount of time, as written on his document. Our project however uses distributed system, i.e. the code for doing the real job are divided in to tasks so that the execution time decreases.

A robotic gripper with variable stiffness and a model for grasping force is proposed by Haibin Yin et al. in 2017. It is verified using a two jawed robotic gripper. The stiffness of gripper varied using the relationship between temperature and elastic modulus of SMA wires. Results show that the grasping ability increases by at least 30% of normal two fingered grippers but the drawback is, it is applicable for only small weights [8]. W. Gauchel has proposed a model individually movable two jawed pneumatic gripper using closed-loop position controller of jaws. But in pneumatic based approach cost and energy consumption is high [23] . Paul Glick et al. have designed a robotic gripper with a combination of fluidic and elastomeric actuators and gecko-inspired adhesives. This has increased the soft gripping property. But the drawback of this technique is position and speed of gripping is complex to control and needs more computation for lifting heavy parts. [10].

K.A Khila and P.S Ampath Umar [11] used technique for controlling an automatic arm utilizing an application build in the android platform. An indication is produced in the android application that will be received through the raspberry pi board and also the automatic arm works according the predefined program, being an android application the command center from the automatic arm. As indicated on the paper the android phone, the application, and the Raspberry are connected to the same Wi-Fi. The motors are being driven by L293D motor driver. Its limitations towards the methodology used is that, it is operated manually and not applicable for industrial applications.

A project proposed by Bernard Franklin, et.al. [12], is not really a humanoid robot but has two 6 DOF robotic arm. But the proposed robot has two robotic arms with 6 Degree of Freedom (DOF), with each arm having two fingered grippers controlled by high torque servo motors. Raspberry Pi and Arduino are used as the controller and robot sensors interface, respectively. The arms are proposed to be controlled by an android application that they have developed using Massachusetts Institute of Technology (MIT) app inventor on-line development tool which is made by MIT students. The proposed methodology is that, an android device sends commands to the Raspberry through Bluetooth but what actual object the robot is going to pick and place is not stated.

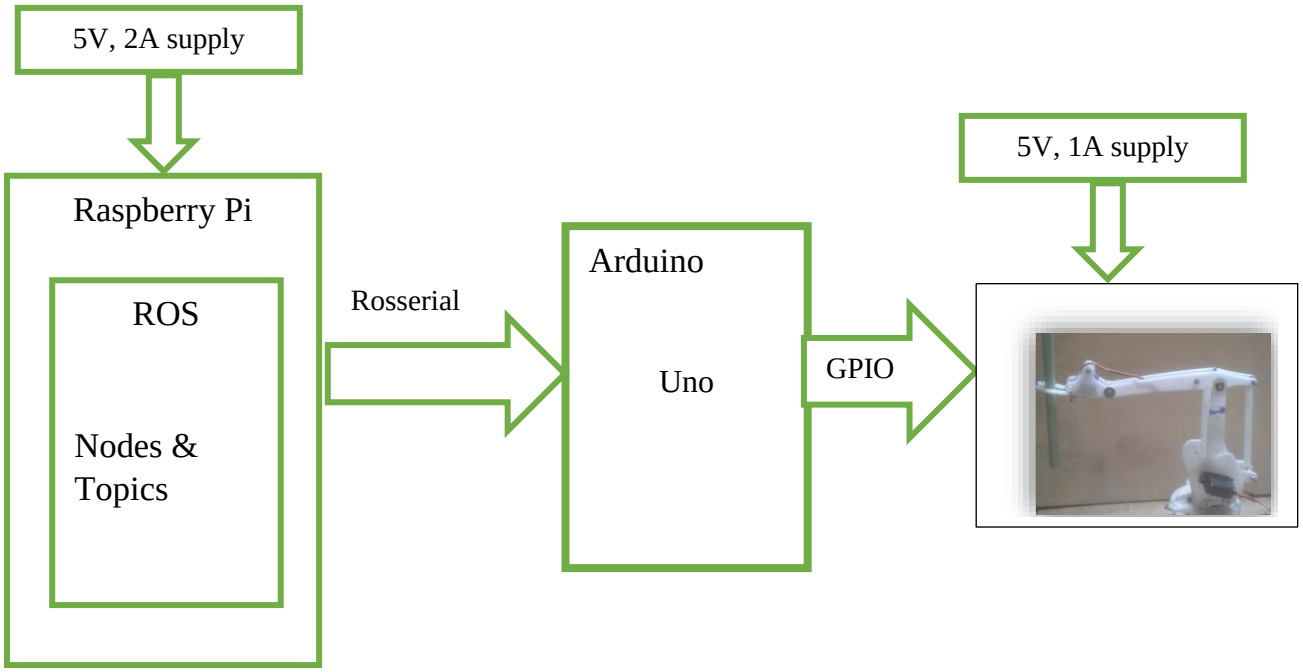
Chapter 3 Methodology

This project is software and hardware design and implementation of a robotic arm and so that, the mathematical aspect. i.e. the robot kinematics, the software design and the hardware arm design are included. The robotic arm is modeled and the forward and inverse kinematics, i.e. the robot kinematics is solved, then to make the robotic arm move as per desired procedures, the software is designed using Robot Operating System (ROS). The hardware design, i.e. selection of hardware components and hacking (modifying) the U-arm robot is done. In this section the overall system block diagram, the robot kinematics and hardware component description are included.

The hardware implementation of this project is done as shown on the block diagram below. The components are selected to make the system stable. Raspberry Pi is as computer and ROS can be installed and programmed on it, and Arduino is good at controlling the servo motors, it has a good library for servo.

3.1. Block diagram

The designed system for this project is shown as the block diagram below.



Raspberry pi is the main controller of the project. Inside the it ROS runs and it controls the system. ROS control mechanism is selected because of its distributed architecture of control. The whole system source code is not at one place, but it divides it into executing nodes that do one specific job. This helps in debugging for errors which may arise and to simply add or remove a process from the system.

Raspberry pi has been selected since it is a credit card sized computer that can run ROS and communicate with peripheral devices like Arduino. It runs on SD-card, for running the OS and processes. This makes handy for robot development. Arduino has been selected for its good library and good control of servo motor.

For the joint motors, servo motor has been selected over stepper motor. This is because servo motor can easily be controlled by sending angle values to it without use of motor driver. The servos in this project are powered from external 5V, 1A supply, Arduino cannot supply a current greater than 40mA, which is not enough for the servos

The communication of Raspberry and Arduino is over serial, using rosserial_arduino library. The low level control like serial communication timing, baudrate, communication port and other serial protocols are all handled by the library.

3.2. Robot Kinematics

Kinematics studies the motion of bodies without consideration of the forces or moments that cause the motion. Robot kinematics refers the analytical study of the motion of a robot manipulator. Formulating the suitable kinematics models for a robot mechanism is very crucial for analyzing the behavior of industrial manipulators. There are mainly two different spaces used in kinematics modelling of manipulators namely, Cartesian space and Quaternion space. The transformation between two Cartesian coordinate systems can be decomposed into a rotation and a translation. There are many ways to represent rotation, including the following: Euler angles, Gibbs vector, Cayley-Klein parameters, Pauli spin matrices, axis and angle, orthonormal matrices, and Hamilton 's quaternions. Of these representations, homogenous transformations based on 4x4 real matrices (orthonormal matrices) have been used most often in robotics. Denavit & Hartenberg (1955) showed that

a general transformation between two joints require four parameters. These parameters known as the Denavit-Hartenberg (DH) parameters have become the standard for describing robot kinematics. Although quaternions constitute an elegant representation for rotation, they have not been used as much as homogenous transformations by the robotics community [27] .

The robot kinematics can be divided into forward kinematics and inverse kinematics. Forward kinematics problem is straightforward and there is no complexity deriving the equations. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. Stated more formally, the forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints.

Hence, there is always a forward kinematics solution of a manipulator. Inverse kinematics is a much more difficult problem than forward kinematics. It is concerned with the inverse problem of finding the joint variables in terms of the end-effector position and orientation. The solution of the inverse kinematics problem is computationally expansive and generally takes a very long time in the real time control of manipulators. Singularities and nonlinearities that may exist when solving for the inverse kinematics for robot arms having more than 2 DOF.

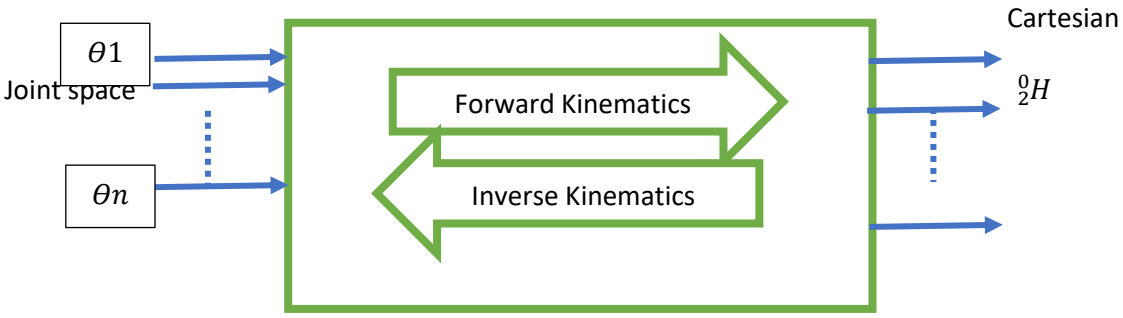


Figure 4: Forward and Inverse kinematic

Two main solution techniques for the inverse kinematics problem are analytical and numerical methods. In the first type, the joint variables are solved analytically according to given configuration data. In the second type of solution, the joint variables are obtained based on the numerical techniques. In this project, the analytical solution of the manipulators is examined rather than numerical solution. There are two approaches in analytical method: geometric and algebraic solutions. Geometric approach is applied to the simple robot structures, such as 2-DOF planar manipulator or less DOF manipulator with parallel joint axes. For the manipulators with more links and whose arms extend into 3 dimensions or more the geometry gets much more tedious. In this case, algebraic approach is more beneficial for the inverse kinematics solution. There are some difficulties to solve the inverse kinematics problem when the kinematics equations are coupled, and multiple solutions and singularities exist. Mathematical solutions for inverse kinematics problem may not always correspond to the physical solutions and method of its solution depends on the robot structure.

3.2.1. Kinematic diagram

Kinematic diagram is a simplified 2D diagram to represent the real object which is in 3D space. That is it helps to visualize an object on a 2D so that, the engineer/designer focuses on the mathematical calculations needed.

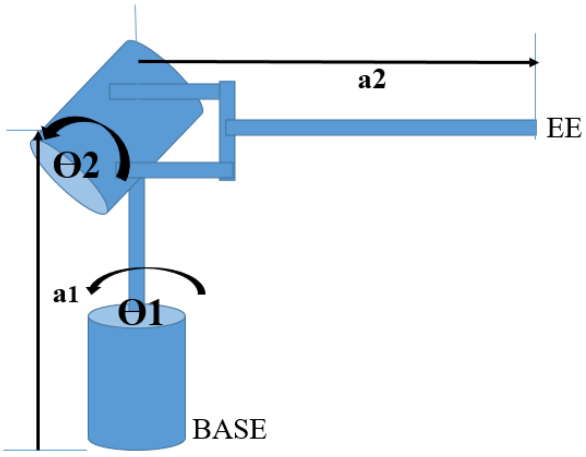


Figure 5: Kinematic Diagram

The base servo motor shaft is normal to the ground and so that it helps the arm to rotate on its axis. The elbow servo motor is designed to make the second link move up and down and so that the rotational axis is perpendicular with the base rotational axis.

3.2.2. Frame Assignment

Frame assignment is based on Denavit-Hartenberg frame assignment rules. There are four rules on how to assign frame on robot to simplify calculation as DH notation.

Rule No 1: Z-axis must be the axis of rotation for revolute joint or axis of translation for prismatic joint.

Rule No 2: X-axis has to be perpendicular to both its own Z-axis and Z-axis before it.

Rule No 3: All frames must follow the right-hand rule notation – for Y-axis assignment.

Rule No 4: Each X-axis must intersect Z-axis of the frame before it.

For convenience, the frame numbers have to be determined first. Let base frame is frame 0, elbow frame is frame 1 and end-effector frame is frame 2. The end-effector (EE) has no joint but, frame must have frame. Doing this helps to know the orientation (rotation) of it with the base frame or elbow frame.

According to rule 1, for base joint, Z-axis (Z0) must be normal to the ground. That is because the joint is revolute joint, i.e. servo motor. For elbow joint (also revolute), Z-axis (Z1) must be towards the negative of the base joint's Y-axis. For the end-effector, the configuration of the frame before it can be used for it, since it doesn't have joint. So, Z-axis (Z2) of EE can be assigned to be in the same direction as Z1.

The base frame X-axis (X0) can be chosen to be in to the page or to the right. But it makes calculation easier to choose to be to the right, since D-H notation uses Z-axis and X-axis. For elbow frame, X1 has to be perpendicular to both Z1 and Z0. So that X1 has to be to the right. The EE frame X2 can be selected to be in the same direction as X1 for the reason mentioned above.

All frames must follow the right-hand rule as D-H rule #3 states. So, Y0 has to be directing in-to the page, Y1 has to be upwards and Y2 has to be also upwards. The kinematic diagram with the frame assignment is shown in the figure below.

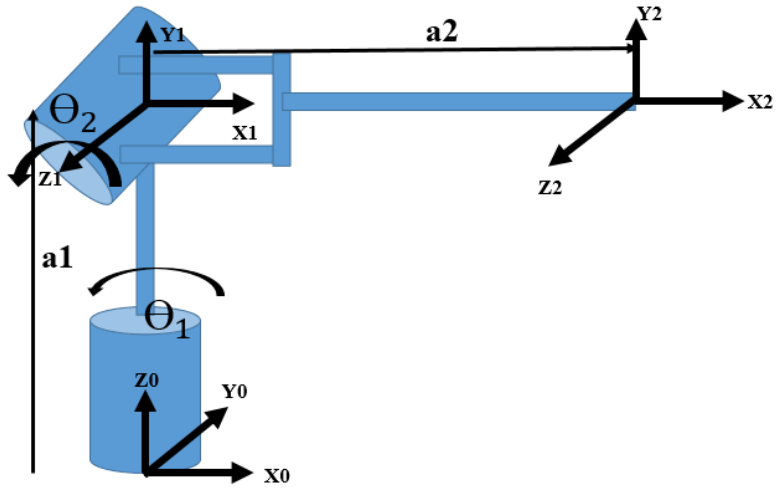


Figure 6: Kinematic diagram with frame

3.2.3. Forward Kinematics

In this project we use the forward kinematics to check whether the robot goes to a specific position when angle is fed. This greatly helps to debug error. The error could be measurement error, the position of robot with respect to the objects, i.e. cylinder and water. The forward kinematics calculation method chosen is the Homogenous Transformation Matrix method. HTM uses rotation matrices and displacement vectors.

3.2.3.1. Rotation Matrix

Rotation matrix constructs the rotation of one frame with the other frame. It is the orientation one frame has with respect to the other frame, which is equivalent with projection of one frame with respect to the other [27] .

The representation going to be used here is ${}^{a}_{b}R$, where a and b are frame numbers for instance, for rotation of frame 1 with respect to 0, ${}^{0}_{1}R$ will be used throughout the paper.

$$ {}^{0}_{1}R = \begin{bmatrix} Projection\ of\ X1\ on\ X0, Y0, Z0 \\ Projection\ of\ Y1\ on\ X0, Y0, Z0 \\ Projection\ of\ Z1\ on\ X0, Y0, Z0 \end{bmatrix}^{T} $$

For an angle theta (Θ) the rotation around X, Y and Z axis is as follows:

$$ R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} $$

$$ R_Y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} $$

$$ R_Z = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

These matrices are found by using projection like mentioned above and when rotating by angle θ.

The orientation of frame 1 with respect to frame 0, is expressed as the rotation matrix of 1 as seen from 0. This matrix is found by multiplying two matrices, rotation matrix of frame 1 from 0 at home position and the rotation matrix of frame 1 from 0 at an angle θ. Mathematically, ${}^{0}_{1}R = {}^{0}_{1}R@home * {}^{0}_{1}R@\theta$ and ${}^{1}_{2}R = {}^{1}_{2}R@home * {}^{1}_{2}R@\theta$

In home position, orientation of frame 1 from 0 and frame 2 from 1 are as expressed below.

$$ {}^{0}_{1}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} $$

$$ {}^{1}_{2}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

Next, ${}^{0}_{1}R@\theta$ and ${}^{1}_{2}R@\theta$ has to be calculated. To calculate these, the direction of rotation has to be known. For ${}^{0}_{1}R$, θ is the rotation around Z0 or around Y1, both of them can be used – both give the same result.

Since D-H frame assignment is used, the rotation will be on its own Z-axis or on Y-axis of the frame after it. When calculating ${}^{0}_{1}R$, the ${}^{0}_{1}R_{,@\theta 1}$ can be selected from rotation around its own Z-axis or around Y-axis of the frame after it. Using rotation around Z-axis,

for ${}^{0}_{1}R$, ${}^{0}_{1}R = R_{Z0,@\theta 1} * {}^{0}_{1}R@home$

$$ {}^{0}_{1}R = \begin{bmatrix} cos\theta 1 & -sin\theta 1 & 0 \\ sin\theta 1 & cos\theta 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} $$

$$ {}^{0}_{1}R = \begin{bmatrix} cos\theta 1 & 0 & sin\theta 1 \\ sin\theta 1 & 0 & cos\theta 1 \\ 0 & 1 & 0 \end{bmatrix} $$

for ${}^{1}_{2}R$, ${}^{1}_{2}R = R_{Z0,@\theta 2} * {}^{1}_{2}R@home$

$$ {}^{1}_{2}R = \begin{bmatrix} cos\theta 2 & -sin\theta 2 & 0 \\ sin\theta 2 & cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}R = \begin{bmatrix} cos\theta 2 & -sin\theta 2 & 0 \\ sin\theta 2 & cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

The orientation of end-effector, i.e. frame 2 from frame 0 is the multiplication of the individual consecutive rotation matrices [13].

$$^{0}_{2}R = {}^{0}_{1}R * {}^{1}_{2}R$$

$$^{0}_{2}R = \begin{bmatrix} cos\theta 1 & 0 & sin\theta 1 \\ sin\theta 1 & 0 & cos\theta 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} cos\theta 2 & -sin\theta 2 & 0 \\ sin\theta 2 & cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^{0}_{2}R = \begin{bmatrix} cos\theta 1 cos\theta 2 & -cos\theta 1 sin\theta 2 & sin\theta 1 \\ sin\theta 1 & -sin\theta 1 sin\theta 2 & -cos\theta 1 \\ sin\theta 2 & cos\theta 2 & 0 \end{bmatrix}$$

3.2.3.2. Displacement Vector

Displacement vector describes the position of one frame with respect to the other frame. When a robot's end-effector moves in space, it has a pose i.e. orientation and position with respect to the base frame. This position is expressed by displacement vector. This vector is represented as $^{a}_{b}d$ where a and b are frame numbers.

The position of frame 1 with respect to frame 0: $^{0}_{1}d = \begin{bmatrix} 0 \\ 0 \\ a1 \end{bmatrix}$, where a1 is the length of link1.

The position of frame 2 with respect to frame 1: $^{1}_{2}d = \begin{bmatrix} a2cos\theta 2 \\ a2sin\theta 2 \\ 0 \end{bmatrix}$, where a2 is the length of second link.

3.2.3.3. Homogenous Transformation Matrix (HTM)

HTM is useful to know the pose, i.e. the orientation and position of the robot as angle is given. The displacement vector cannot be multiplied as the rotation matrices to have the end-effector position with respect to the base frame. But the HTM can be multiplied like the rotation matrices. It is represented as $^{a}_{b}H$, where a and b are frame numbers.

The calculation for HTM is as follows,

$$^{0}_{1}H = {}^{0}_{1}R * {}^{0}_{1}d$$

$$^{1}_{2}H = {}^{1}_{2}R * {}^{1}_{2}d$$

and the overall HTM is: $$^{0}_{2}H = {}^{0}_{1}H * {}^{1}_{2}H$$

$$\begin{bmatrix} R & R & R| & d \\ R & R & R| & d \\ R & R & R| & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

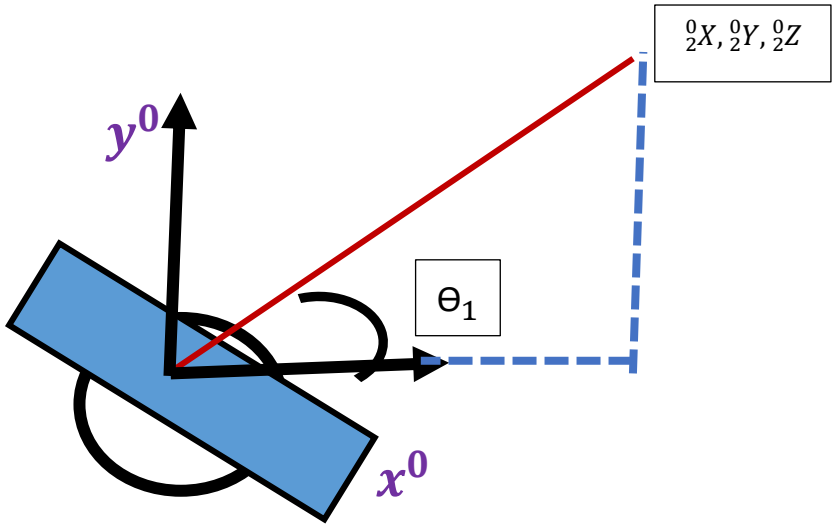
After expressing the individual equations, finding the HTM can be done by computer. Using python language, it makes the calculation easier. The python code is pasted under Appendix.

In the code NumPy library is used for matrix calculations and math library is used for trigonometric calculations

3.2.4. Inverse Kinematics

In this project analytical Inverse Kinematics solving is used. Analytical calculation decreases the complexity of the calculation.

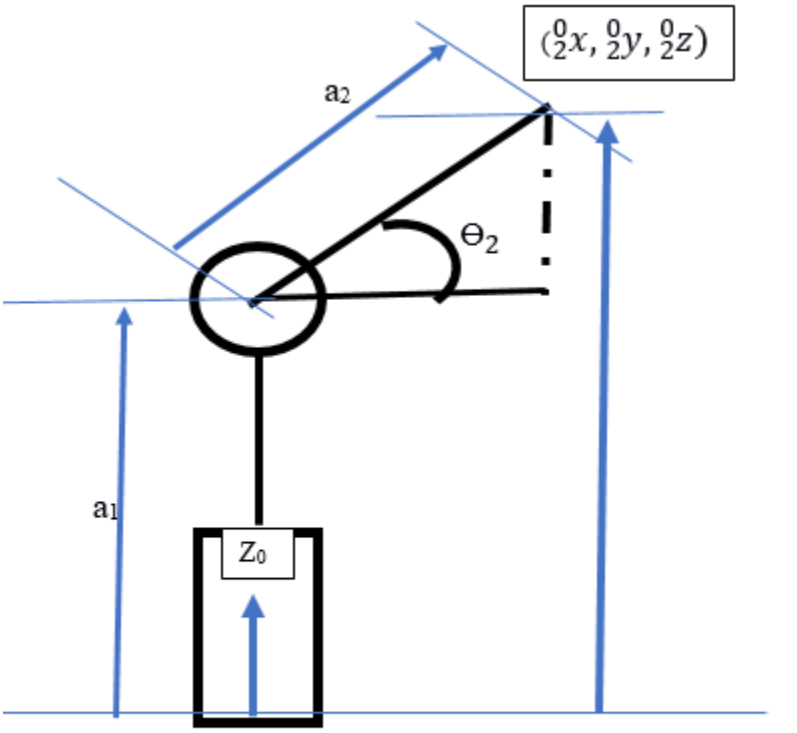
To solve for Θ_1: Top view of the robot can be used as shown below:



$$\Theta_1 = \tan^{-1} \frac{^{0}_{2}y}{^{0}_{2}x}$$

Figure 7: Top view of the arm

To solve for Θ_2: Side view of the robot can be used which is as shown below:



$$\Theta_2 = \tan^{-1}\left(\frac{{}^0_2Z - a_1}{\sqrt{{a_2}^2 - ({}^0_2Z - {a_1}^2)^2}}\right)$$

Figure 8: Side view of the arm

3.3. Hardware component description

3.3.1. Raspberry Pi

The Raspberry Pi sprang out of a desire by colleagues at the University of Cambridge's Computer Laboratory to see a return to the days of kids programming inexpensive computers. The rise of expensive PCs and games consoles put paid to the BBC B, Spectrum and C64 generation of home programmers, leading to applicants for computer studies

courses lacking the necessary skills. After spending a few years designing prototypes, Eben Upton, formerly of the University but now working as a chip designer for Broadcom, joined forces with his old university colleagues, Pete Lomas of hardware design company Norcott Technologies and David Braben, co-author of classic BBC Micro game Elite, to form the Raspberry Foundation. Three years later, the Raspberry Pi went into mass production with the Model B (main image), and then later, lower-capacity RAM version, but cheaper, Model A.

The name, Raspberry Pi, was the combination of the desire to create an alternative fruit-based computer (such as Apple, BlackBerry, and Apricot) and a nod to the original concept of a simple computer that can be programmed using Python (shortened to Pi) [14].

The beauty of the Raspberry Pi is that it's just a very tiny general-purpose computer (which may be a little slower than normal PC but much better at some other stuff than a regular PC), so you can do anything you could do on a regular computer with it. In addition, the Raspberry Pi has powerful multimedia and 3D graphics capabilities, so it has the potential to be used as a game's platform [28]

The processor at the heart of the Raspberry Pi system is a Broadcom BCM2835 system-on-chip (SoC) multimedia processor. This means that the vast majority of the system's components, including its central and graphics processing units along with the audio and communications hardware, are built onto that single component hidden beneath the 256 MB memory chip at the center of the board. [28]

The basic concepts behind the Raspberry Pi were to make it as affordable as possible, supply only the basics and provide it with a programming environment and hardware connections for electronics projects. The Pi runs a modified version of Linux called Raspian with Wheezy Raspian being the preferred option for newcomers to the device. Raspian runs directly on an SD card and provides a command-line interface for using the operating system. However, as this was likely to be a little overwhelming for novices, there's a more friendly face to the Pi, and that appears when you type "StartX" to launch the desktop.

As mentioned, the Pi comes as a bare bones circuit board but you can buy all the extras needed to make it fully functional. The first thing that is needed is an SD card to act as storage for the operating system and any software you want to install. Although it will work on a 4GB card, it is recommended to use an 8GB card as a minimum. This should be Class 10 speed and it is better to have a branded card so that it is more reliable.

The Raspberry Pi requires a 5V power supply, which can comfortably supply at least 700mA (1A or more recommended) with a micro USB connection. It is possible to power the unit using portable battery packs, such as the ones suitable for powering or recharging tablets. Again, ensure that they can supply 5V at 700mA or over.

The HDMI connection also includes audio so if you are using it with a TV or a monitor with speakers then sound will come from those. If not, there's a separate 3.5mm audio jack for output. The final option is that on the Model B there is an Ethernet connection for wiring into internet modem. This is a faster and more hassle-free way of getting the Pi online.

Nowadays, there are many OSes that can run on Raspberry Pi. Raspbian is the recommended OS by Raspberry Pi foundation. Many Linux flavors like Ubuntu, Arch Linux, Ubuntu Mate, RISC OS and also Windows 10 for Raspberry is also available!

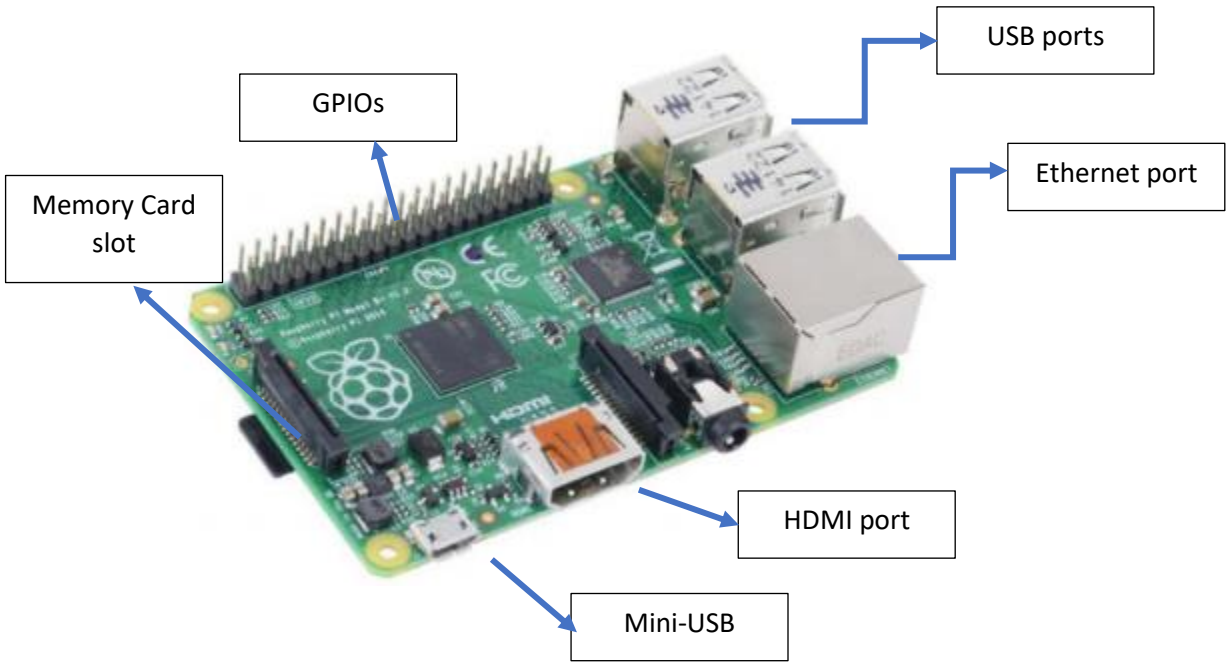


Figure 9: Raspberry Pi

The Raspberry Pi is available in two models- Model A and Model B. The Model A is cheaper compared to Model B but lack some connectors. It was initially started with the intention of teaching the basic computer science in schools but later turned out to be a wonder in the field of single board computers. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The user can attempt overclocking, up to 1 GHz, without affecting the warranty), VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage. The Broadcom BCM2835 incorporates an ARM1176 processor core. The ARM processors are low-cost yet high performance processors used in smart phones, digital TVs, eReaders and other media devices. ARM licenses the processor to other companies, like Broadcom, who combine it with various I/O modules and incorporate it into system-on-chip designs. ARM, also provides the physical IP for the digital inputs and outputs (GPIO) as cell libraries.

3.3.2. Arduino

Arduino is an open source project consisting of both hardware and software. It was originally created to give designers and artists a prototyping platform for interaction design courses. Today hobbyists and experts all over the world use it to create physical computing projects. After releasing several Arduino boards and Arduino IDE versions, the Arduino team decided to specify a version 1.0 of the platform. It will be the reference for all future developments, and they announced it on the first day of 2010. Since then, they have released the Arduino Uno, and they have also improved the IDE and its supporting libraries step-by-step [15 1]

Over the years the Arduino team improved the board's design and released several new versions. They usually had Italian names such as Uno, Duemilanove, or Diecimila, and you can find a list of all boards that were ever created by the Arduino team online [15].

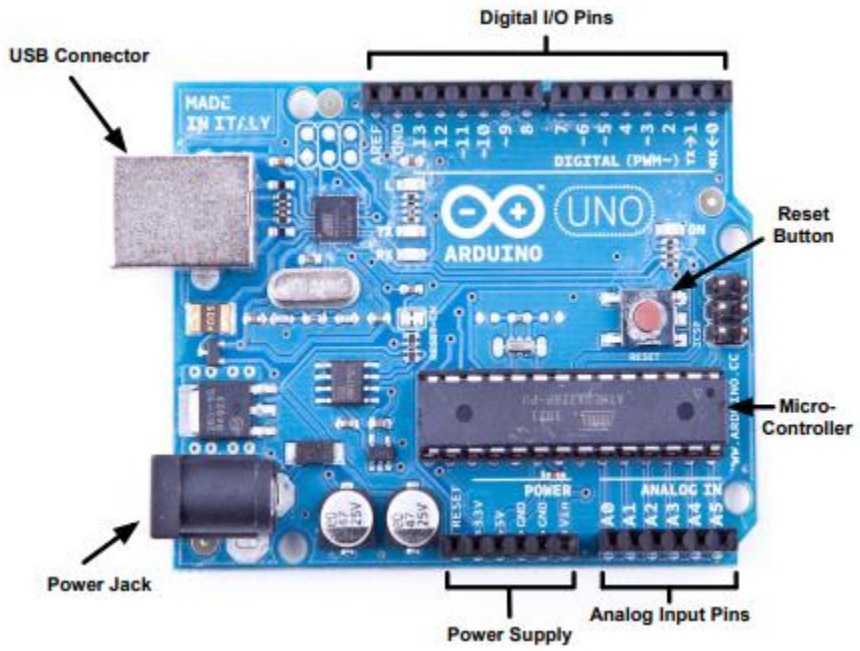


Figure 10: Arduino

As an electronic device, the Arduino needs power. One way to power it is to connect it to a computer's USB port, but that isn't a good solution in some cases. Some projects don't necessarily need a computer, and it would be overkill to use a whole computer just to power the Arduino. At the board's top are 14 digital IO pins named D0–D13. Depending on the

need, these pins can be used for both digital input and output. Six of them (D3, D5, D6, D9, D10, and D11) can also act as analog.

The Arduino board is a small microcontroller board, which is a small circuit (the board) that contains a whole computer on a small chip (the microcontroller). This computer is at least a thousand times less powerful than the MacBook I'm using to write this, but it's a lot cheaper and very useful to build interesting devices. Look at the Arduino board: you'll see a black chip with 28 "legs"—that chip is the ATmega328, the heart of your board.

Specifications

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended):7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328)
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

The Arduino software, known as the Integrated Development Environment (IDE), is free. It can be downloaded from www.arduino.cc. The Arduino IDE is based on the Processing language, which was developed to help artists create computer art without having to first become software engineers. The Arduino IDE can run on Windows, Macintosh, and Linux.

The Arduino board is inexpensive (about $35) and quite tolerant of common novice mistakes. Because the Arduino hardware and software are open source, the Arduino hardware design itself can be downloaded and the software code is also downloadable – because it is open source [16].

The Arduino microcontroller was originally created as an educational platform for a class project at the Interaction Design Institute Ivrea in 2005. It grew from the previous work of the Wiring microcontroller designed by Hernando Barragán in 2004. 23 From its inception, the Arduino was developed to engage artistic and design-oriented minds [17].

3.4. Materials Data from the Factory

Prior to selecting the robotic arm mechanical structure, the factory data on the working environment has been collected. The following data are collected that help for this project

- The test cylinder is 21 cm height with 1cm diameter
- The water outlet position is at x= 18.0 cm
- The test cylinder has to be placed back to the prior position

Chapter 4 Software Design

As introduced earlier, the robotic arm is designed for picking the test cylinder, going to predetermined position and placing it. The predetermined position where the robot goes after it picks up the cylinder is a water outlet position. For simplicity of explanation and code, water is used for water outlet position and cylinder is used for test cylinder position.

4.1. System Flow Chart

The robot is designed to pick a cylinder, go to water outlet position (water) and place the cylinder back. Thus, five states are designed. They are home position, cylinder position, picking, water outlet position and placing. The flow chart indicating the flow of software design is shown below.

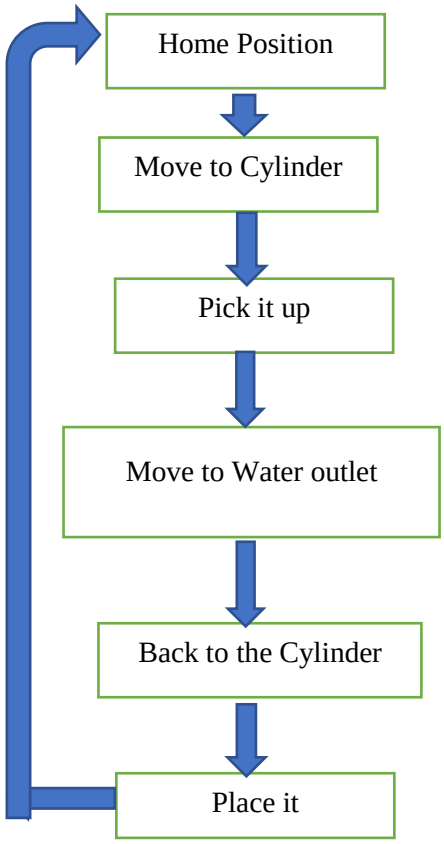


Figure 11: Flow char of the system

Home position is where the robot stays until the user starts the robot. And also, after it has placed the cylinder, it goes to home position and stays here again. Cylinder position is where

the cylinder to be picked is placed. As the user starts the robot, it automatically goes to this position. After it has gone to cylinder it picks it up and then goes to water to wash the cylinder. Washing the cylinder is designed to just stay in water outlet position for 3 seconds. After 3 seconds, it goes back to the cylinder and then it places it and goes to home position and stays there.

In the ROS architecture designed for this project, Home Position, going to cylinder, picking up, moving to water outlet, moving back to cylinder and placing are assigned a particular node named home, s_cylinder, e_cylinder, pick, water and place. s_cylinder and e_cylinder are named like this to indicate the task of going to cylinder from home position and task of going to cylinder from water outlet position, i.e. to cylinder at start and to cylinder at the end, respectively.

To start the robot the user has to type start. Immediately after the command it solves inverse kinematics for cylinder position and goes there. It adjusts its angle so that it can go to cylinder to pick, as it reaches the cylinder, the gripper servo grips the cylinder. Solving of inverse kinematics is also done for water position

The flow chart of each node is shown below ROS uses distributed system. For this project also there is one control node that controls the current and the future process. The flow chart for it is also shown.

Flow Chart for each node

HOME

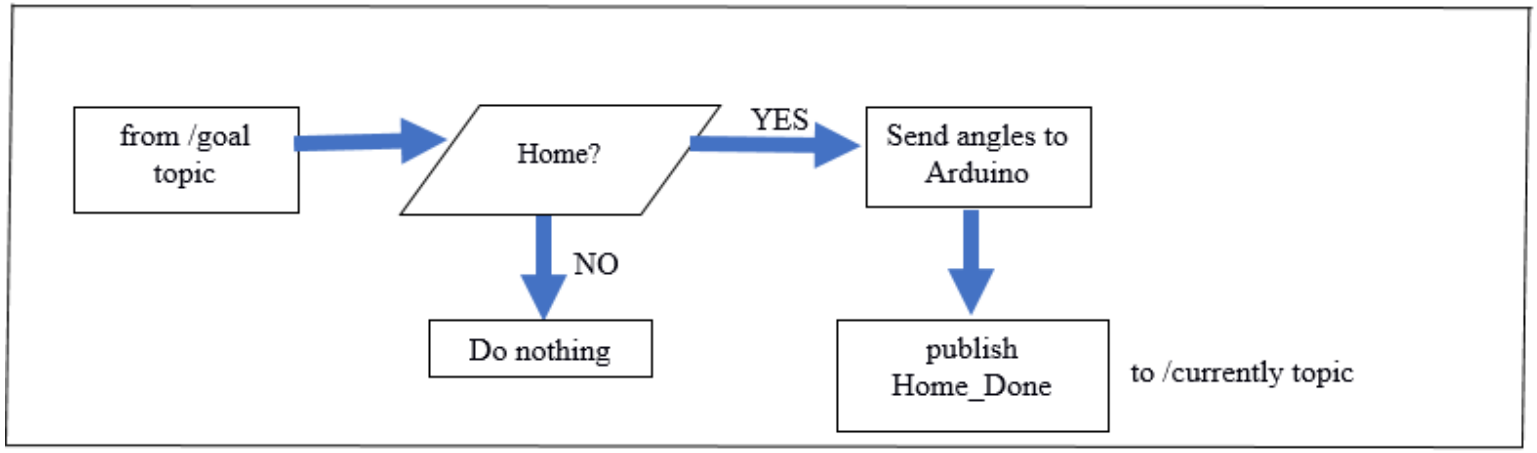


Figure 12: Home node flow chart

S_CYLINDER

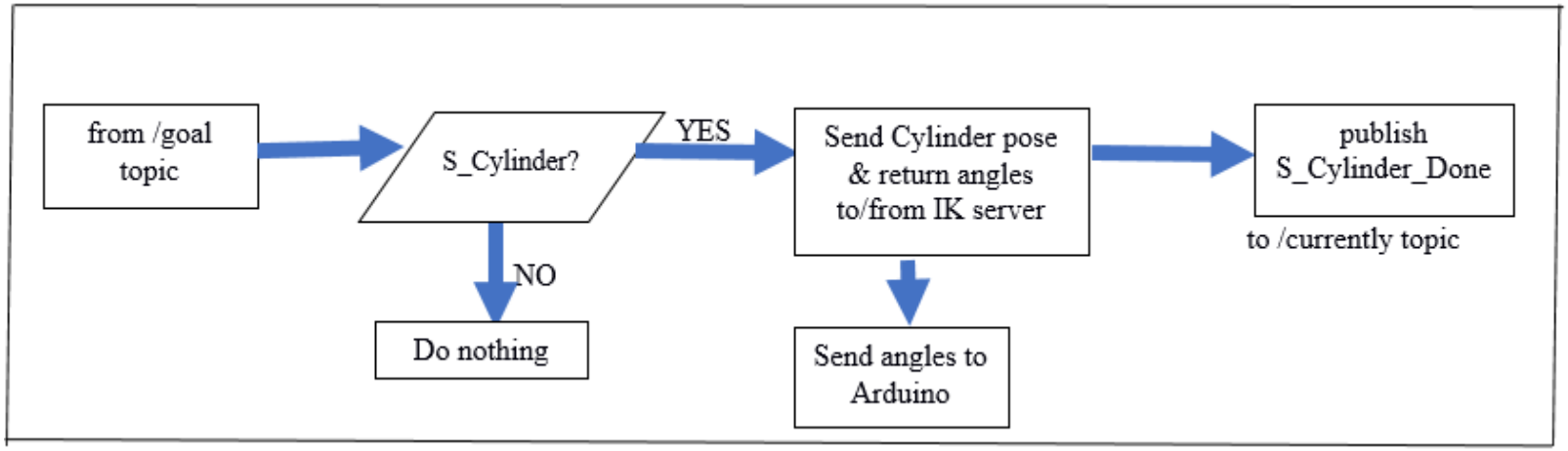


Figure 13: S_Cylinder flow chart

E_CYLINDER

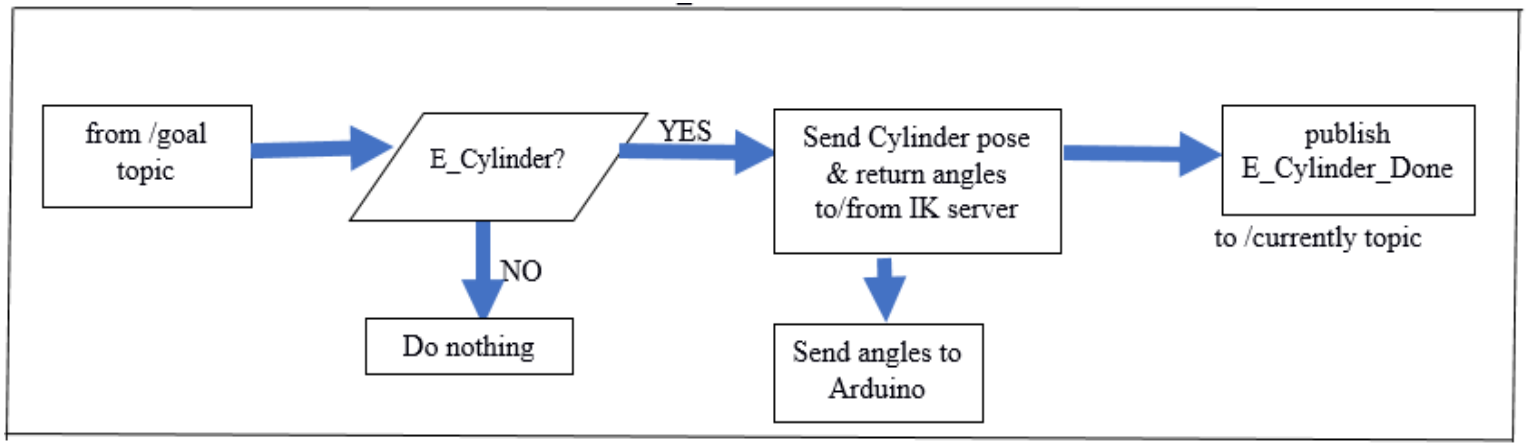


Figure 14: E_Cylinder node flow chart

PICK

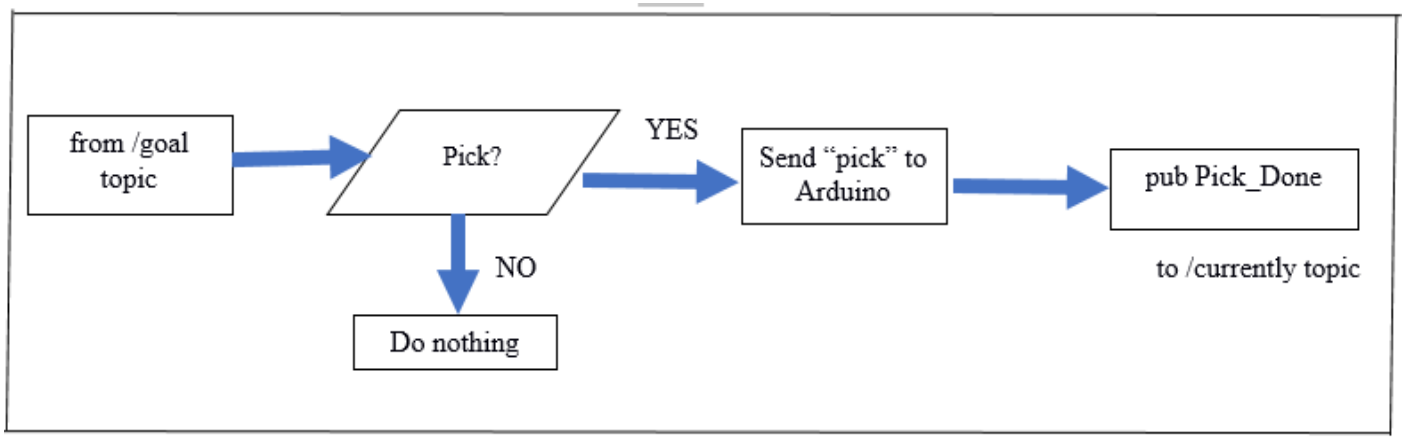


Figure 15:Pick node flow chart

WATER

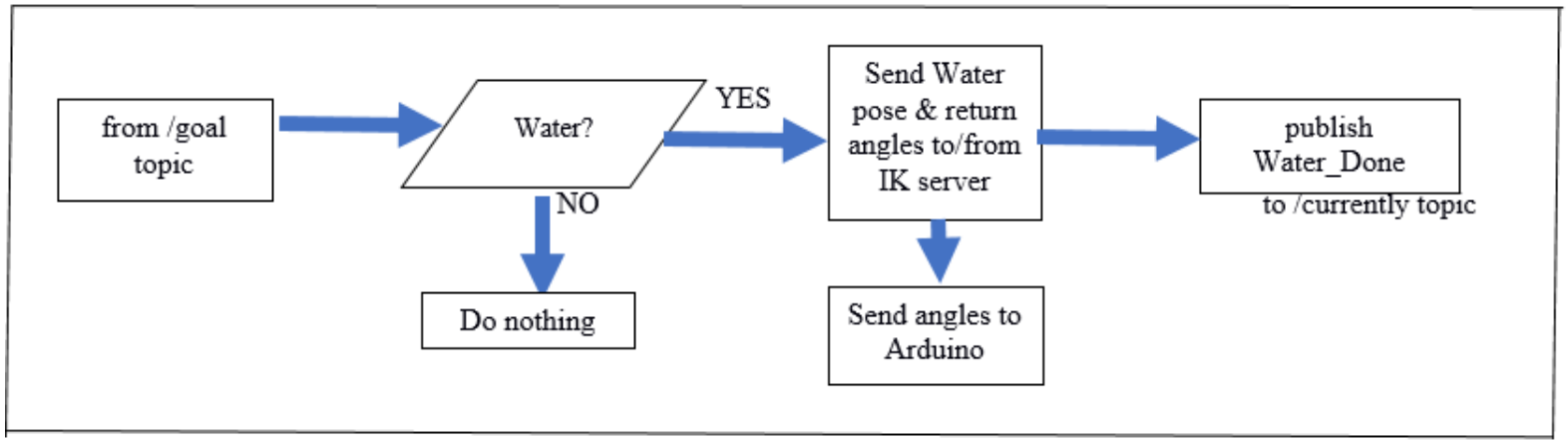


Figure 16: Water node flow chart

PLACE

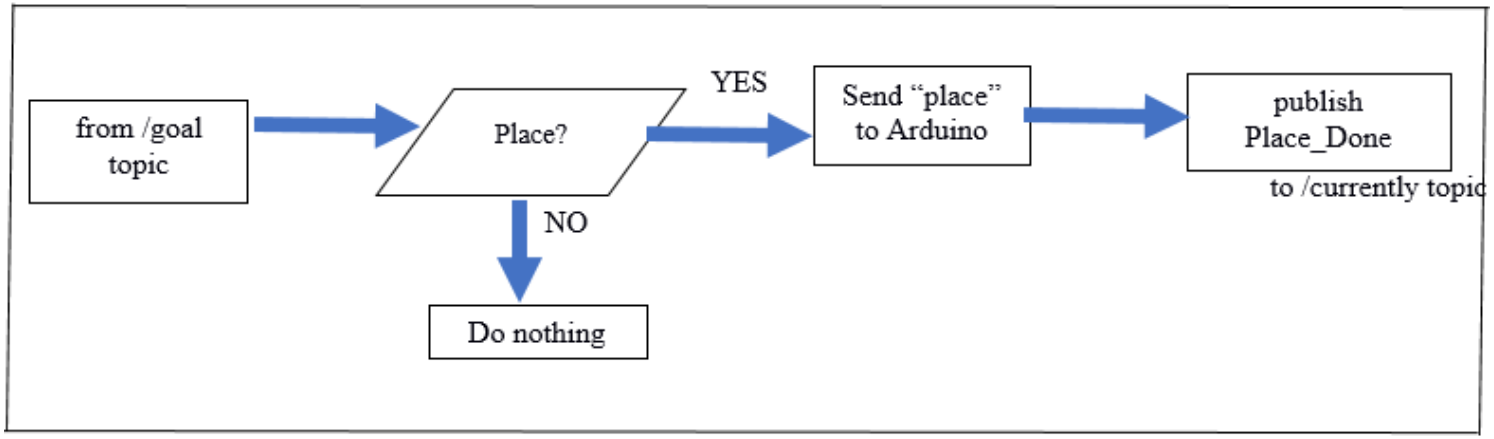


Figure 17: Place node flow chart

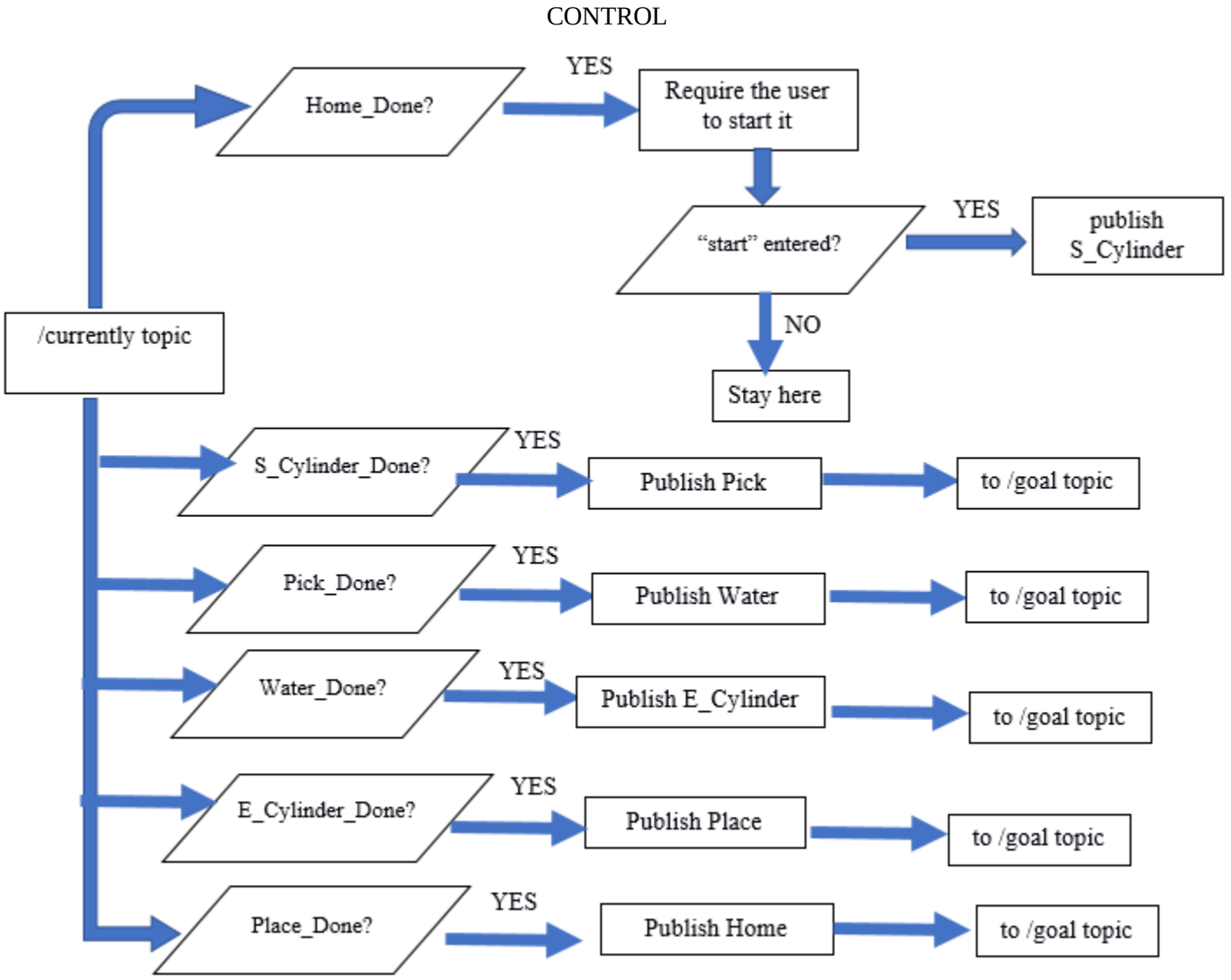


Figure 18: Control node flow chart

4.2. ROS Programming

Programming in ROS is somewhat different from other programming styles in that it uses other languages to create it own. As explained under software component description section (pp), ROS uses three client libraries rospy, roscpp and roslisp. Since it uses distributed system there are many components like nodes, services, launch files, action files etc. These files are individual source codes which has to be saved in particular folders. The source files for nodes and services are paced in src folder of the package created, launch file is launch folder and service file in srv folder.

The python client library rospy is used for the source files of nodes and service. The source files of nodes, services, launch and others are placed in a workspace called catkin_ws. Inside the workspace, a package called thesisproject.is created. The python codes for each node is written inside src folder of the package, src is to mean source. The service file (which is IK.srv) is in srv folder, launch file, which is robotarm.launch, is in launch folder of the package. The steps followed to create workspace and package are written under Appendix.

Launch file is used to start multiple nodes at the same time. It is written in Extensible Markup Language (XML) format. In this project there are eight (8) nodes that has to be run in parallel. To make this happen, launch file called robotarm.launch is created and XML code is written to start all nodes at the same time. When launch file is executed it looks for nodes that are named and executes them. For this purpose, nodes are made executable using chmod +x <nodename.py> on Linux terminal, where nodename.py represents the python file for the node. The steps followed to create the launch file and the code inside of it is written under Appendix.

4.2.1. Home Node

Home position is where the robot stays as it is in standby position. To start the robot, the user has to enter a start command then the robot starts to go through the flow chart. This is done when the home node gets Home message from /goal topic. As the user enters start command, this home node sends angle to Arduino through serial. The angle to be sent are

‘0.0 160.0’. The first number is for the base joint, the second number is for the elbow joint. After angles are sent this node publishes Home_Done on /currently topic.

4.2.2. S_Cylinder Node

Cylinder position is where the cylinder to be picked up is placed. The cylinder is assumed to be placed at predetermined position. This position is going to be sent for inverse kinematic solver service and it returns the two angles, i.e. the base and elbow angles, to make the gripper reach the cylinder. The gripper servo at this time has to be opened, facilitating for picking. These two angles are sent to Arduino through serial and after that this node publishers S_Cylinder_Done on /currently topic.

4.2.3. Pick Node

Picking is the action of changing the gripper angle so that the gripper grips the cylinder. This process is controlled by pick node. As it finds Pick command from /goal topic, it sends “pick” (string) message to Arduino and the Arduino changes the angle of the gripper servo to 45 degrees. The gripper is made of finger like things which are connected by gear. So, when the servo rotates one of the fingers the other rotates too.

After the node sends pick message to Arduino, it publishes Pick_Done on /currently topic.

4.2.4. Water Node

Water position is a position where water outlet is placed. Water outlet is water continuously poured from its source, so it is used to clean the test cylinder after the examination is completed. The procedures the robot does in this project is in post-examination. After the completion, it picks, cleans and places back.

In this watering position it stays for seconds. The node for this process is called water. This node sends water outlet position to inverse kinematic solver service and accepts angle from it. Then it sends these angles, i.e. the base and elbow joint angles, to Arduino through serial. After that it publishes Water_Done on /currently topic

4.2.5. E_Cylinder Node

This node is to be called after water node finishes, i.e. to facilitate for placing of the cylinder. Like the S_Cylinder node, when it is called, it sends cylinder position to Inverse Kinematics solver service and gets the base and elbow angles. Then it sends these angels to Arduino through serial. Finally, it publishes E_Cylinder_Done to /currently topic.

4.2.6. Place Node

Placing node is for placing the cylinder back to its prior position. This is the opposite of picking. The node for this process is called place. As it receives place command from /goal topic, it sends place message to Arduino and the Arduino changes the gripper servo angle to 90. In this process the gripper releases the cylinder and the cylinder is placed! Finally, this node publishes Place_Done on /currently topic

4.2.7. Control node

Control node is used to control the flow of the process. It controls the sequence of processes. It subscribes to /currently topic and publishes to /goal topic. It uses simple if-else statement to check in what stage the robot is depending on the message on /currently topic then it publishes the next procedure on /goal topic. As explained under each node, they subscribe to /goal topic and do their respective job.

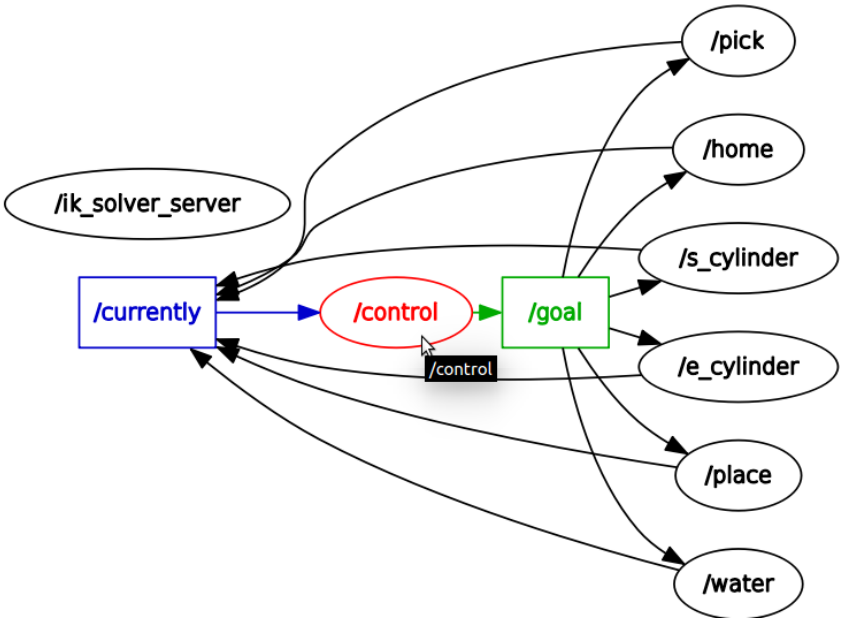


Figure 19: Connection of all nodes, generated by rqt_graph

Names with circle is nodes and names with rectangle is a topic. For instance, water, place are nodes and currently and goal are topics.

4.2.8. IK Solver Server

This node is a server node for solving inverse kinematics. It takes the three position values and returns base joint angle (theta1) and elbow joint angle (theta2). These angles are calculated as the calculations under inverse kinematics section. The node for this service is called /ik_solver_server. This node initializes the inverse kinematic solving service. It is called only when needed. There are steps to setup service in ROS. The steps are covered in ROS programming section.

4.3. Software component description

4.3.1. Robot Operating System (ROS)

4.3.1.1 Introduction

ROS (pronounced as "Ross"), the Robot Operating System, is an open source framework for getting robots to do things. ROS is meant to serve as a common software platform for people who are building and using robots. This common platform lets people share code and ideas more readily and, perhaps more importantly, means that everyone does not have to spend years writing software infrastructure before the robots start moving! [26] It provides the services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. [1]

Since December 2015 only, in the official distribution of ROS, there are over 2,000 software packages of ROS, written and maintained by almost 600 people. Approximately 80 commercially available robots are supported, and there are least 1,850 academic papers that mention ROS. So, one no longer have to write everything from scratch, especially if we're working with one of the many robots that support ROS.[26]

The primary goal of ROS is to support code *reuse* in robotics research and development. ROS is a distributed framework of processes (aka *Nodes*) that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into *Packages* and *Stacks*, which can be easily shared and distributed. ROS also supports a federated system of code *Repositories* that enable collaboration to be distributed as well. This design, from the file-system level to the community level, enables independent decisions about development and implementation, but all can be brought together with ROS infrastructure tools. [1] Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator, and other messages.

ROS is not a real-time framework, though it is possible to integrate ROS with Realtime code. The Willow Garage PR2 robot uses a system called pr2_etherCAT, which transports ROS messages in and out of a real-time process. ROS also has seamless integration with the Orocos Real-time Toolkit. [1]

There are three main ROS client libraries, roscpp – ROS with C++, rospy – ROS with python and roslisp – ROS with lisp. These main ROS client libraries (C++, Python, and Lisp) are geared toward a Unix-like system, primarily because of their dependence on large collections of open-source software dependencies. For these client libraries, Ubuntu Linux is listed as "Supported" while other variants such as Fedora Linux, macOS, and Microsoft Windows are designated "Experimental" and are supported by the community.[2]

There is also an Android client library for ROS. It is called rosjava. rosjava has also enabled ROS to be integrated into an officially supported MATLAB toolbox which can be used on Linux, macOS, and Microsoft Windows.[29]

A JavaScript client library, roslibjs has also been developed which enables integration of software into a ROS system via any standards-compliant web browser [20]. In September 2018 Microsoft ported Core ROS to Windows 10. [5]

4.3.1.2 History

ROS was first started by PhD students Eric Berger and Keenan Wyrobek at Stanford robotics laboratory. While working on robots to do manipulation tasks in human environments, the two students noticed that many of their colleagues were held back by the diverse nature of robotics: an excellent software developer might not have the hardware knowledge required, someone developing state of the art path planning might not know how to do the computer vision required. In an attempt to remedy this situation, the two students set out to make a baseline system that would provide a starting place for others in academia to build upon. In the words of Eric Berger, "something that didn't suck, in all of those different dimensions" [30] . In their first steps towards this unifying system, the two build the PR1 robot as a hardware prototype and began to work on software from it, borrowing the best practices from other early open source robotic software frameworks [21] .

Willow Garage began developing the PR2 robot as a follow-up to the PR1, and ROS as the software to run it. Groups from more than twenty institutions made contributions to ROS, both the core software and the growing number of packages which worked with ROS to form a greater software ecosystem [22] . Willow Garage began 2012 by creating the Open Source Robotics Foundation (OSRF) [41] in April. The OSRF was immediately awarded a software contract by DARPA [23] . On September 1st, 2014, NASA announced the first robot to run ROS in space: Robotnaut 2, on the International Space Station.[24] . In 2017, the OSRF changed its name to Open Robotics.

4.3.1.3 Design of ROS

The philosophy behind ROS is that it was designed with open-source in mind, intending that users would be able to choose the configuration of tools and libraries which interacted with the core of ROS, so that users could shift their software stacks to fit their robot and application area. As such, there is very little which is actually core to ROS, beyond the general structure within which programs must exist and communicate.

ROS processes are represented as nodes in a graph structure, connected by edges called topics.[25] ROS nodes can pass messages to one another through topics, make service calls to other nodes, provide a service for other nodes, or set or retrieve shared data from a communal database called the parameter server. A process called the ROS Master makes all of this possible by registering nodes to itself, setting up node-to-node communication for topics, and controlling parameter server updates. Messages and service calls do not pass through the master, rather the master sets up peer-to-peer communication between all node processes after they register themselves with the master. This decentralized architecture lends itself well to robots, which often consist of a subset of networked computer hardware, and may communicate with off-board computers for heavy computation or commands.

4.3.1.3.1. Nodes

A node represents a single process running the ROS graph. Every node has a name, which it registers with the ROS master before it can take any other actions. Multiple nodes with different names can exist under different namespaces, or a node can be defined as anonymous, in which case it will randomly generate an additional identifier to add to its given name. Nodes are at the center of ROS programming, as most ROS client code is in the form of a ROS node which takes actions based on information received from other nodes, sends information to other nodes, or sends and receives requests for actions to and from other nodes.

4.3.1.3.2. Topics

Topics are named buses over which nodes send and receive messages.[25] Topic names must be unique within their namespace as well. To send messages to a topic, a node must publish to said topic, while to receive messages it must subscribe. The publish/subscribe model is anonymous: no node knows which nodes are sending or receiving on a topic, only that it is sending/receiving on that topic. The types of messages passed on a topic vary widely and can be user-defined. The content of these messages can be sensor data, motor control commands, state information, actuator commands, or anything else

.

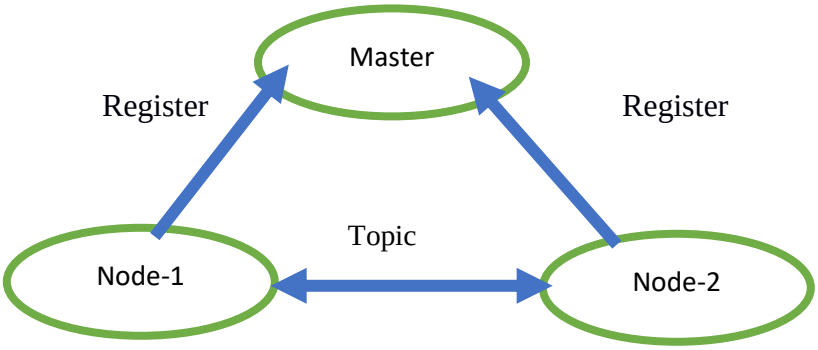


Figure 20: ROS Communication

As figure above shows, all nodes have to register to a master and once register the master takes care of the connection between nodes. The topics carry the data published or the data that is to be subscribed or to take from.

4.3.1.3.3. Services

A node may also advertise services [27] . A service represents an action that a node can take which will have a single result. Services have input and output argument. They are not available always or always publishing or subscribing but they are available for only when called. As such, services are often used for actions which have a defined beginning and end, such as capturing a single-frame image, rather than processing velocity commands to a wheel motor or odometer data from a wheel encoder. Nodes advertise services and call services from one another.

4.3.1.3.4. Tools

ROS's core functionality is augmented by a variety of tools which allow developers to visualize and record data, easily navigate the ROS package structures, and create scripts automating complex configuration and setup processes. The addition of these tools greatly increases the capabilities of systems using ROS by simplifying and providing solutions to a number of common robotics development. These tools are provided in packages like any other algorithm, but rather than providing implementations of hardware drivers or algorithms

for various robotic tasks, these packages provide task and robot-agnostic tools which come with the core of most modern ROS installations.

4.3.1.3.5. catkin

catkin [28] is the ROS build system, having replaced rosbuild [29] as of ROS Groovy. catkin is based on CMake, and is similarly cross-platform, open source, and language-independent. Catkin is the way ROS makes building and interconnection easy.

4.3.1.3.6. roslaunch

roslaunch [73] is a tool used to launch multiple ROS nodes both locally or remotely. roslaunch configuration files, which are written using XML can easily automate a complex startup and configuration process into a single command. roslaunch scripts can include other roslaunch scripts, launch nodes on specific machines, and even restart processes which die during execution.

Chapter 5 Result and Discussion

5.1. Result

The roslaunch thesisproject robotarm.launch is excuted at the terminal, all nodes are registered at the master. As start is entered at the terminal the robot has started to go to the cylinder, closed the gripper, picked the cylinder, gone to water position, came back to cylinder, opened the gripper, placed it and gone to home and stayed there, respectively.

The robot arm first starts in home position as shown below.

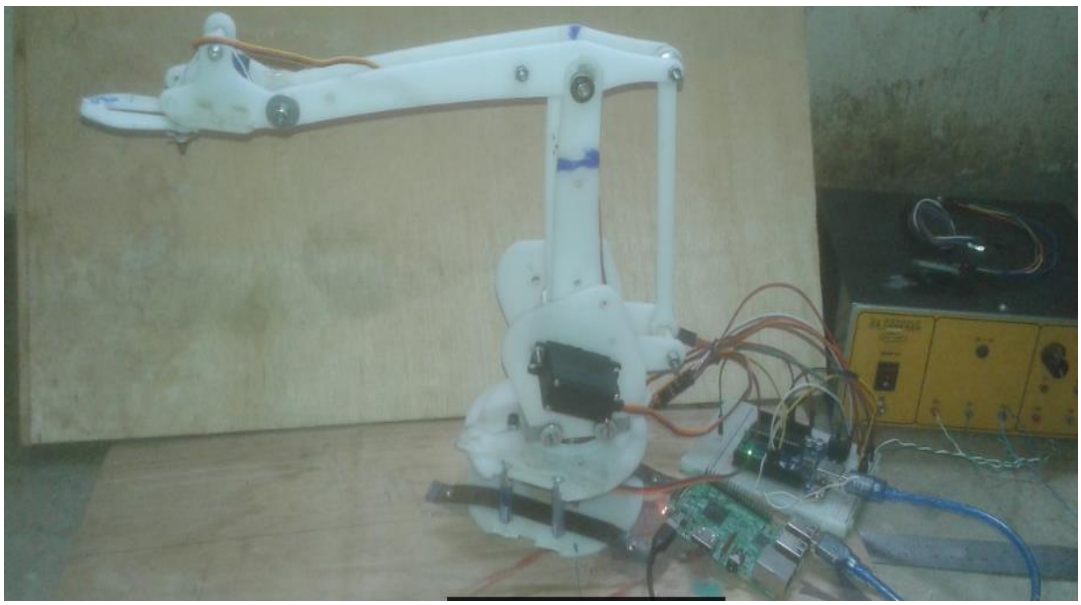


Figure 21: The robot in Home position

Then it goes to cylinder:

As the time of writing and until document submission, the test cylinder cannot be found so a tube is used instead, since it does not make a difference.

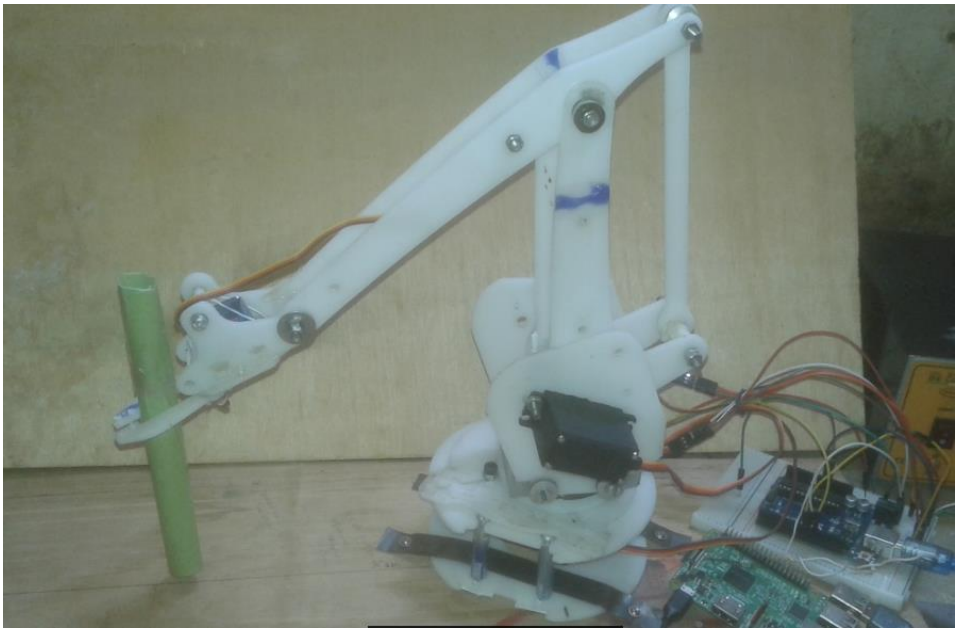


Figure 22: The robot in cylinder position

Picking the cylinder, gripping it:

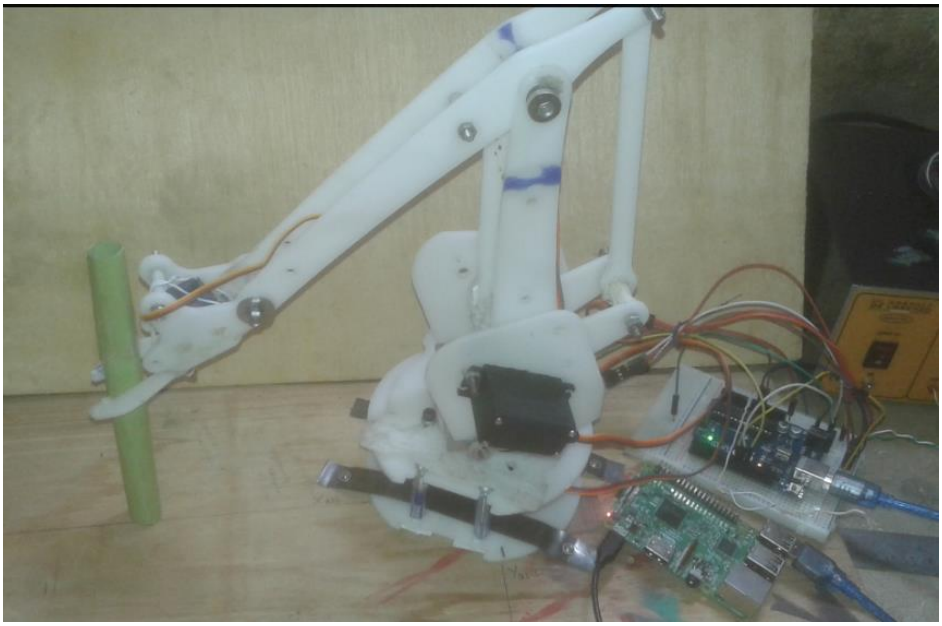


Figure 23: Picking

Water outlet position:

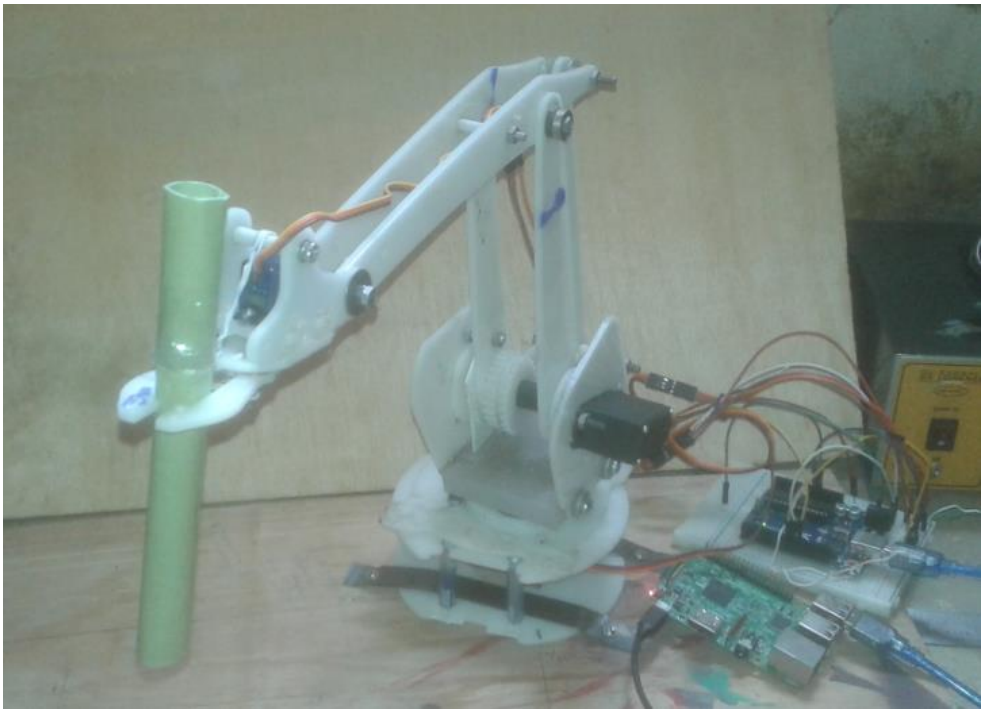


Figure 24: The robot in Water position

Placing:

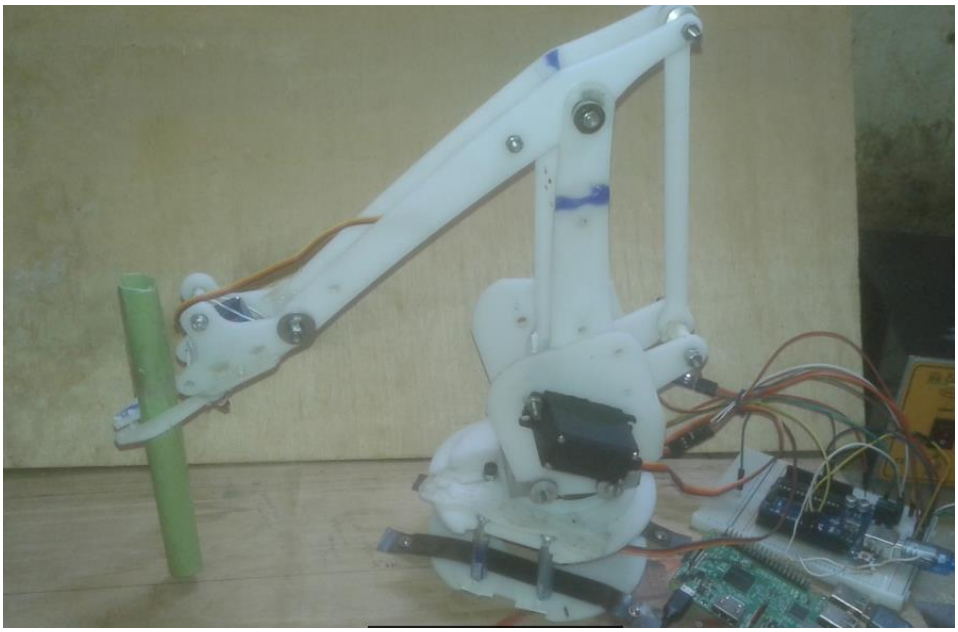


Figure 25: Placing

5.2. Discussion

As the robot executes the code and move the servos, the stages at which the robot currently was viewed using rostopic echo /goal. This command shows the goal set by the control node. It sequentially goes from Home to place.

Figure 21, show as the robot is in home position. the robot stayed in this position until start is entered. This position is when the joint angle are zero. But in the hardware, the zero position goes to the origin of frame 0, which is not the same as the kinematic diagram. To make it similar to the kinematic diagram, base joint angle is set to zero and elbow joint angle is set to 150 degrees. The inverse kinematics returns angles assuming both joint angles are at zero in home position and feeding the angle directly to the servos does not get the end-effector. To get this right, the elbow joint angle that is written to the servo is made 150+theta.T2, where theta.T2 is the angle from inverse kinematics.

Figure 22 shows the robot in cylinder, as it goes to pick the cylinder. When the cylinder goes to this stage, the gripper is opened so that, the gripper does not break or hit the cylinder.

Figure 23 shows that the gripper has picked the cylinder. That is the gripper changes angle so that it can grip it.

Figure 24 shows that the robot has gone to water outlet position. Although there is no water outlet on the picture, it is assumed that there is a water outlet. The robot stayed here for three (3) seconds.

Figure 25 shows the robot placing. Placing is done by changing the angle of gripper servo so that it releases it. After this position the robot goes to home position without changing the gripper servo angle.

The s_cylinder, e_cylinder and water nodes used the inverse kinematic solver server during the execution. During the execution, there is a delay between the procedures, this could be due to the serial communication or Arduino’s speed of interpretation of the data taken through the serial.

As the code is written for one cycle execution, the user has to enter start again to repeat the process. The Arduino was really good at changing the angle of the servos and staying at the position for a specified time.

Chapter 6 Conclusion and Recommendation

6.1. Conclusion

A pick and place robots usually have few degrees of freedom and with less computational time. The programming and simulation may be simple but not the implementation. In this project, robotic arm which can pick and place a test cylinder is designed and implemented. The stages it followed were staying at home position until user starts it, going to cylinder, picking the test cylinder, going to water outlet and staying for 3 seconds, going back to cylinder and placing it. After it places it returns to home position and waits for command from user.

The software frame work used is ROS and eight nodes have been created. The inverse kinematic solver server was also set to be called when inverse kinematics is needed. Raspberry pi, small computer, is used as the controller through the installed ROS and Arduino has been used to actuate the servos.

The home position used in calculation and actual hardware robot differs in the elbow joint value. This was addressed by adding the home hardware home pisiton and the angle from the inverse kinematics.

The robot arm implemented in this project is not fully satisfying as desired but it is a really good starting point. The reason of this is the hardware design of the robot arm. The working of the robot is like articulated manipulators and the kinematics also follows like that, but on implementation, it causes restriction on elbow join angles. Because of this limitation, the angle sent to Arduino is mapped for the elbow joint servo motor in this project.

6.2. Recommendation

The robotic arm hardware used, i.e. U-arm has faced errors in actuating to the right position with right angle. This design is used in this project, since it was the last design the printing guys can do in this country. So, using an articulated type of robot is recommended since there is no need to map.

The code written for the arm helps do through the stages/procedures, but if interrupted it has to be started again. This problem is not addressed in this project since this needs serious hardware programming and no sufficient time for it. For future works it is recommended to include memory to continuously save the current stage of the robot.

As the gripper sometimes could firmly grip and hold the cylinder, it will be better if force sensor is attached to it.

Reference

[1] Industrial Robots and Robot System Safety, *https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html*

[2] *"ROS/Installation - ROS Wiki"*. Wiki.ros.org. Accessed 13 April 2019

[3] D. Chikurtev, I. Rangelov, N. Chivarov , E. Markov , K. Yovchev, *"Control of Robotic Arm Manipulator Using ROS",* European Polytechnical University – Pernik, May 2018, pp 52-59

[4] Ashly Baby , Chinnu Augustine, Chinnu Thampi, Maria George , Abhilash A P ,Philip C Jose, *"Pick and Place Robotic Arm Implementation Using Arduino",* IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), 2320-3331, Volume 12, Issue 2 Ver. III (Mar. – Apr. 2017), pp 38-41

[5] Priyambada Mishra, Riki Patel, Trushit Upadhyaya, Arpan Desai, *"Development of Robotic Arm Using Arduino Uno",* International Journal on Recent Research in Science, Engineering & Technology (JRRSET), 2348-3105 (online), May 2017

[6] Joseph-Engelberger, "*Unimate - The First Industrial Robot*," https://www.robotics.org/joseph-engelberger/unimate.cfm. , Accessed April 15, 2019

[7] A. Sinha, R. K. Mishra, and S. Jaiswal, "*Robust and Smooth Nonlinear Control of an Industrial Robot for Automated Pick and Place*", International Conference on Computing Communication Control and Automation, 2015.

[8] H. Yin, X. Zhang, J. Li, and J. Cao, "*Grasping model and experiment of a soft robot gripper with variable stiffness*", 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE

[9] W. Gauchel and R. Schell, "*Control of a Servo-pneumatic Gripper with Individually Movable Jaws*", IEEE International Symposium on Intelligent Control, 2006.

[10] Glick, P., Suresh, S., Ruffatto, D., Cutkosky, M., Tolley, M. and Parness, A.. "*A Soft Robotic Gripper With Gecko-Inspired Adhesive*". IEEE Robotics and Automation Letters, 3(2), 2018, pp.903-910.

[11] K.A Khila, P.S Ampath K Umar, "*Smart Phone Based Robotic Arm Control using Raspberry Pi and Wi-Fi*", International Journal of Advanced Technology and Innovative Research (IJATIR), 2348–2370, August-2016,

[12] Bernard Franklin , Sachin.P, Jagadish.S, Shaista Noor , Rajashekhar C. Biradar , "*Pick And Place Humanoid Robot Using Raspberry Pi And Arduino For Industrial Applications",* International Journal of Advance Research in Science and Engineering (IJARSE), 2319-8354, April 2018

[26] Morgan Quigley, Brian Gerkey, and William D. Smart *"Programming Robots with ROS",*

[13] Serdar Kucuk and Zafer Bingul, "*Industrial-Robotics-Theory-Modelling-Control".* ARS/plV, Germany, December 2006, pp. 964,

[14] Simon Monk*, "Raspberry Pi Cookbook",* USA, O'Reilly Media, Inc, 2014, pp 1-5

[15] Maik Schmidt, "*Arduino A Quick-Start Guide",* APress, 2013, pp 123

[16] Massimo Banzi, *"Getting Started with Arduino"*, 2nd ed, Packt pub, 2015, pp 235-237

[17] "*Interaction Design Institute Ivrea"* http://interactionivrea.org/en/index.asp. The Academy. Accessed November 19, 2009

[18] Aaron Asadi *, "Raspberry Pi The Complete Manual", 2014 pp 8-10*

[19] "Robot Operating System (ROS) Support from MATLAB - Hardware Support". Mathworks.com. Accessed 6 May 2019.

[20] *"roslibjs - ROS Wiki". wiki.ros.org. Retrieved 29 April 2019.*

[21] *"STAIR". stair.Stanford.edu. Retrieved 12 December 2017.*

[22] *"Repositories". ROS.org. Retrieved 7 June 2011.*

[23] *"DARPA Awards Simulation Software Contract to Open Source Robotics Foundation".* https://spectrum.ieee.org/automaton/robotics/robotics-software/darpa-robotics-challenge-simulation-software-open-source-robotics-foundation, Accessed 17 April 2019

[24] *"ROS running on ISS - ROS robotics news".* www.ROS.org. Accessed 12 May 2019.

[25] *"ROS/Tutorials/UnderstandingNodes - ROS Wiki".* wiki.ros.org. Accessed 29 April 2019.

[27] *"ROS/Tutorials/UnderstandingServicesParams - ROS Wiki".* wiki.ros.org. Accessed 29 April 2019.

[28] *"catkin - ROS Wiki". wiki.ros.org.* Accessed *29* April 2019.

[29] *"rosbuild - ROS Wiki". wiki.ros.org.* Accessed 29 April 2019.

[30] Guizzo, Evan Ackerman and Erico *"Wizards of ROS: Willow Garage and the Making of the Robot Operating System".* IEEE Spectrum: Technology, Engineering, and Science News. Accessed 29 April 2019.

Appendix

Python code for Homogenous Transformation Matrix

```
### the code starts here###

import numpy as np

import math


T1 = 0  # Theta 1 in degrees

T2 = 0  # Theta 2 in degrees


a1 = 27.5 # link 1 in cm

a2 = 22.5

#     ------- Forward Kinematics -------

T1 = math.radians(T1)# Changing to radians

T2 = math.radians(T2)

# Rotation Matrix

R0_1 = np.array([[math.cos(T1), 0, math.sin(T1)], [math.sin(T1), 0, -math.cos(T1)], [0, 1, 0]])

R1_2 = np.array([[math.cos(T2), -math.sin(T2), 0], [math.sin(T2), math.cos(T2), 0], [0, 0, 1]])

R0_2 = np.dot(R0_1, R1_2)

# displacement vector

d0_1 = np.array([[0], [0], [a1]])

d1_2 = np.array([[a2 * math.cos(T2)], [a2 * math.sin(T2)], [0]])

# HTM - Homogenous Transformation Matrix

H0_1 = np.concatenate((R0_1, d0_1), axis=1)

H0_1 = np.concatenate((H0_1, [[0, 0, 0, 1]]), axis=0)

H1_2 = np.concatenate((R1_2, d1_2), axis=1)

H1_2 = np.concatenate((H1_2, [[0, 0, 0, 1]]), axis=0)

H0_2 = np.dot(H0_1, H1_2)

print("\n\n", np.matrix(H0_2))
```

Creating Workspace and Package

a. mkdir -p ~/catkin_ws/src
b. cd ~/catkin_ws/src
c. catkin_init_workspace # to initialize the workspace
d. cd ~/catkin_ws
e. catkin_make # this command builds the package
f. cd ~/catkin_ws/src
g. catkin_create_pkg thesisproject rospy std_msgs
h. cd ~/catkin_ws && catkin_make

Generating service file

a. mkdir ~/catkin_ws/src/thesisproject/srv
b. nano ~/catkin_ws/src/thesisproject/srv/IK.srv
c. write: *float32 x0_2*

float32 y0_2

float32 z0_2

---

float32 T1

float32 T2

The first three are inputs, the three dash lines (---) describes the end of input and the last two are outputs. This helps for the server to know it has three inputs and two outputs with their respective names.

d. Inside package.xml:

<build_depend>message_generation</build_depend> and *<exec_depend>message_runtime</exec_depend>* are added so that catkin knows that it has to generate a message during build and use the message during runtime.

e. In CMakeLists.txt:
    - In *find_package* function *message_generation* is included at the end but inside the brackets
    - Under *add_service_files* function *IK.srv* is added inside the brackets. IK.srv is the service file created as in *srv* folder of the package

- Under *generate_messages* function added *std_msgs* as message generation dependency.

f. cd ~/catkin_ws/ && catkin_make

code for ROS nodes

control node

```
#! /usr/bin/env python

import rospy

from std_msgs.msg import String

import time

import sys

def command(msg):

    if msg.data=='Home_Done':

            print 'The Robot is in rest condition, you can start it by typing < start >'

            # if start is entered--pub.publish('Cylinder')

            #pub.publish('S_Cylinder')

            #while msg.data=='Home_Done':

                    #print 'going to Cylinder'

            time.sleep(2)

    elif msg.data=='S_Cylinder_Done':

            pub.publish('Pick') # sending Pick to /goal topic

            while msg.data=='S_Cylinder_Done':

                    time.sleep(2)

            elif msg.data=='Pick_Done':

            pub.publish('Water')    # sending Water to /goal topic

            while msg.data=='Pick_Done':

                    time.sleep(2)

    elif msg.data== 'Water_Done':


            pub.publish('E_Cylinder')    # sending Cylinder2 to /goal topic

            while msg.data=='Water_Done':

                    time.sleep(2)

    elif msg.data=='E_Cylinder_Done':

            pub.publish('Place')    # sending Place to /goal topic

            while msg.data=='E_Cylinder_Done':

                    time.sleep(2)

    elif msg.data=='Place_Done':

            pub.publish('Home')    # sending Home to /goal topic

            while msg.data=='Place_Done':
```

```
            time.sleep(2)

if __name__=="__main__":

    rospy.init_node('contol')

    rate=rospy.Rate(2)

    pub=rospy.Publisher('goal',  String,
queue_size=1)
```

Home node

```
#! /usr/bin/env python


import rospy

import serial

from std_msgs.msg import String

from time import sleep

# from thesisproject.srv import IK  ## not
important here


def go_home(msg):

    if msg.data=='Home':


            sleep(2)

            pub.publish('Home_Done')


if __name__=="__main__":

    rospy.init_node('home')

    rate=rospy.Rate(2)


    pub=rospy.Publisher('currently',      String,
queue_size=1)

    while not rospy.is_shutdown():


    sub=rospy.Subscriber('currently',
String, command)

            rate.sleep()



    while not rospy.is_shutdown():

            sub=rospy.Subscriber('goal',
String, go_home)

            rate.sleep()
```

S_Cylinder node

```
#! /usr/bin/env python


import rospy

import serial

from std_msgs.msg import String

from thesisproject.srv import IK

from time import sleep

def go_to_cylinder(msg):


    # cylinder position

    x0_2=1.1

    y0_2=2.1

    z0_2=3.1


    if msg.data=='S_Cylinder':


    rospy.wait_for_service('ik_solver')
```

```
    service=rospy.ServiceProxy('ik_solver', IK)


                theta=service(x0_2,y0_2,z0_2)

                # the pricise value of their value respectively

                # use theta.T1 and theta.T2



                # set the pwm for the angles to go to the cylinder to be picked up



                # send this to arduino serial

                #ser.write('%d %d %d' %(theta.T1, theta.T2))

                #print theta.T1, theta.T2

                #print '\n going to cylinder'

                sleep(2)

                pub.publish('S_Cylinder_Done')

if __name__=="__main__":


    rospy.init_node('s_cylinder')

    rate=rospy.Rate(2)


    pub=rospy.Publisher('currently', String, queue_size=1)



    #ser=serial.Serial('/dev/ttyACM0',9600)

    while not rospy.is_shutdown():

                sub=rospy.Subscriber('goal', String, go_to_cylinder)

                rate.sleep()
```

Pick node

```
#! /usr/bin/env python


import rospy

import serial

from std_msgs.msg import String

from time import sleep


def go_pick(msg):

    if msg.data=='Pick':

                sleep(2)

                pub.publish('Pick_Done')

if __name__=="__main__":

    rospy.init_node('pick')

    rate=rospy.Rate(2)


    pub=rospy.Publisher('currently', String, queue_size=1)


    while not rospy.is_shutdown():

                sub=rospy.Subscriber('goal', String, go_pick)

                rate.sleep()
```

Water node

```
#! /usr/bin/env python
```

```
import rospy

import serial

from std_msgs.msg import String

from time import sleep

from thesisproject.srv import IK

def go_to_water(msg):

    # cylinder position

    x0_2=1.1

    y0_2=2.1

    z0_2=3.1

    if msg.data=='Water':

    rospy.wait_for_service('ik_solver')
service=rospy.ServiceProxy('ik_solver', IK)

                theta=service(x0_2,y0_2,z0_2)

                # returning array from a function

                 sleep(2)

                pub.publish('Water_Done')

if __name__=="__main__":

    rospy.init_node('water')

    rate=rospy.Rate(2)

    pub=rospy.Publisher('currently',      String,
queue_size=1)

    #ser=serial.Serial('/dev/ttyACM0',9600)

    while not rospy.is_shutdown():

                sub=rospy.Subscriber('goal',
String, go_to_water)

                rate.sleep()
```

E_Cylinder node

```
    #! /usr/bin/env python

import rospy

from std_msgs.msg import String

from std_msgs.msg import Float32MultiArray

from thesisproject.srv import IK

from time import sleep

def go_to_cylinder(msg):

    # cylinder position

    x0_2=10.0

    y0_2=6.1

    z0_2=11.1

    if msg.data=='E_Cylinder':

    rospy.wait_for_service('ik_solver')

    service=rospy.ServiceProxy('ik_solver',
IK)

                theta=service(x0_2,y0_2,z0_2)

                t.data=[theta.T1,theta.T2]

                pub_angle.publish(t)

                sleep(2)
    pub_done.publish('E_Cylinder_Done')



if __name__=="__main__":

    t=Float32MultiArray ()

     t.data = []

    rospy.init_node('e_cylinder')

    rate=rospy.Rate(2)

    pub_angle = rospy.Publisher('angles',
Float32MultiArray, queue_size=1)

    pub_done=rospy.Publisher('currently',
String, queue_size=1)

    while not rospy.is_shutdown():
```

```
        sub=rospy.Subscriber('goal', String, go_to_cylinder)

        rate.sleep()
```

Place node

```
#! /usr/bin/env python

import rospy

import serial

from std_msgs.msg import String

from time import sleep

def go_place(msg):

    #global pub

    if msg.data=='Place':

        sleep(2)

        pub.publish('Place_Done')

if __name__=="__main__":

    rospy.init_node('place')

    rate=rospy.Rate(2)

    pub=rospy.Publisher('currently', String, queue_size=1)



    while not rospy.is_shutdown():

        sub=rospy.Subscriber('goal', String, go_place)

        rate.sleep()
```

IK Server node

```
#! /usr/bin/env python

import rospy

import math

from thesisproject.srv import IK

#this is a server for ik solving

#the idea is that it will be called as a node wants

def ik_solver(req):

    T1=T2=0.0

    a1=27.5

    a2=22.5

    T1=math.degrees(math.atan(req.y0_2/req.x0_2))

    T2=math.degrees(math.atan((req.z0_2-a1)/(math.sqrt((a2*a2)-(req.z0_2-a1*a1)))))

    t=[T1,T2]

    return t

def ik_solver_server():

    rospy.init_node('ik_solver_server')

    s=rospy.Service('ik_solver', IK, ik_solver)

    rospy.spin()

if __name__=="__main__":

    ik_solver_server()
```