Tampere University

Muhammad Junaid Iqbal

COMMISSIONING AND SYSTEM INTEGRATION TESTS FOR AN INDUSTRIAL MANIPULATOR WORKSTATION

Faculty of Engineering and
Natural Sciences
Master's Thesis
November 2019

ABSTRACT

Muhammad Junaid Iqbal: Commissioning and System Integration Tests for an Industrial Manipulator Workstation
Master of Science Thesis
Tampere University
MSc. Automation Engineering
November 2019

Industrial systems are composed of several sub systems and architectures that are provided by different manufacturers. System integration aims at enabling a developer to combine these unit systems with limited functionality into one system that can accomplish the execution of required process. Modern integrated systems are developed on top of service-oriented architecture and use webservices for information exchange. Such systems are swiftly deployable and ensure platform interoperability, system adaptability and service reusability. Meanwhile, system integration tests help to reduce the complexity during the integration phase thus ensuring process uniformity.

This thesis focuses on deploying a robotic manipulator in an industrial cell. The robot is installed in the assembly line as service provider while services are invoked by using RESTful web services. Second objective of the thesis is to implement a free shape path planning algorithm for the deployed autonomous manipulator to follow the desired curve. The last component of this thesis is focused on developing integration tests to examine and verify the designed system.

The robot was commissioned at the FASTory assembly line, installed at FAST lab of Tampere University. The free shape paths were implemented by interpolating Bezier curves using De Casteljau algorithm. System was successfully integrated and verified using Top-down depth first and bottom-up breadth first integration testing approaches.

Keywords: System Integration, Integration Tests, Web services, De Casteljau algorithm, Free shape algorithms, Bezier Curves, SCARA robot, Anthropomorphic robot.

PREFACE

I would like to thank Prof. Dr Jose Martinez Lastra for giving me an opportunity to be a student of automation engineering at Tampere University and for teaching and giving us valuable knowledge in the field. I would also like to thank Mr. Luis Gonzalez Moctezuma and Dr Borja Ramis Ferrer for teaching and helping with automation concepts and courses. I extend my grateful thanks to Prof. Jose M. Lastra and Luis G. Moctezuma for supervising this thesis, and for guiding me through the project and providing valuable inputs. Special thanks to Miika Suomalainen and Olatz De Miguel, they have been a great help for this project. I also wish to thank all my colleagues, helping staff at Fast Lab and all the authors and editors referenced in this thesis.

Tampere, 03 November 2019

Junaid Iqbal

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ACE	Automation Control Environment
aibo	Artificial Intelligence Robot
Aka	Also known as
AMF	American Machine and Foundry
API	Application Programming Interface
ARC	Advanced REST Client
ARP	Address Resolution Protocol
BFS	Breadth First Search
CPS	Cyber Physical Devices
DARPA	Defence Advanced Research Projects Agency (U.S.)
DCS	Distributed Control Systems
DFS	Depth First Search
DOF	Degree of Freedom
EsaLAN	Elan Safety Local Area Network
FIFO	First in First out
FPGA	Field-programmable Gate Array
FTP	File Transfer Protocol
GM	General Motors
HMI	Human Machine Interface
HP	Hewlett-Packard
HTTP	Hypertext Transfer Protocol
IBVS	Image-Based Visual Servo
ICMP	Internet Control message Protocol
IFR	International Federation of Robotics
IL	Internet Layer
IOT	Internet of Things
IP	Internet Protocol
IPR	Ingress Protection Rating
IR	Industrial Robot
IR	Industrial Robot
ISO	International Standards Organization
ISS	International Space Station
JIRA	Japanese Industrial Robot Association
JSON	JavaScript Object Notation
KRL	KUKA Robot Language
KSS	KUKA System Software
LLC	Logic Link Control
MAC	Media Access Control
MCU	Micro Control Unit
MES	Manufacturing execution systems
MITI	Ministry of International Trade and Industry (Japan)
MTU	Maximum Transfer Unit
NASA	National Aeronautics and Space Administration
NL	Network Layer
OSI	Open Systems Interconnection
POP3	Post Office Protocol version 3
PUMA	Programmable Universal Machine for Assembly
PVBS	Position-Based Visual Servo
R.U. R	Rossum’s Universal Robots
RARP	Reverse Address Resolution Protocol
REST	Representational State Transfer

RFID	Radio Frequency Identification
RIA	Robotics Institute of America
RPC	Remote procedure call
RTU	Remote Terminal Unit
SCARA	Selective Compliance Assembly Robot
SIT	System Integration Testing
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TL	Transport Layer
U.S.	United States
UDP	User Datagram Protocol
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WS	Web Services
WS	Workstation
XML	Extensible Markup Language

1. INTRODUCTION

1.1 Background

According to International Federation of Robotics (IFR), 384,000 robots were shipped worldwide only in 2018. IFR predicts that only in 2021, 630,000 robots will be dispatched around the globe.[1] It is estimated that by 2020, robotics market will grow to $40 billion. As Robotic applications are increasing on industry floor, manufacturing systems are becoming more reliable and efficient. Manufacturing system's Stability and precision are quite important factors. Because of advantages associated with using a robotic arm in industrial process, these manipulators are being used in helping workers lift and move equipment and material. They also find their applications in tasks like surgeries, pressing, welding and drawing.

Adaptability, reusability and interoperability are the key features of any manufacturing system. Exchange of information between different devices helps in building an intelligent system and a better integrated system. Service reusability and interoperability are functionalities that make Service-Oriented architecture (SOA) an essential tool in system development. Web services (WS) feature language transparency and they are a realization of SOA. Language transparency makes machine-to-machine communication, over a network, language independent. Hence, web services allow devices or applications from different sources to communicate. WS support Remote Procedure Calls (RPCs) enabling a client to invoke services.

System Integration Testing (SIT) is an important part of system development process. It helps verify a system's behaviour, interaction between the modules. SIT makes sure that errors are detected at early stages and makes it easier to fix those errors in integrated systems. Basic objective of integration testing is to help in developing an error free, working version of a system, where modules are interacting in required manners, data follow and memory allocations are working as expected.

1.2 Problem Statement

Demand for innovative products has made the manufacturing processes more complex. Processes within a factory are comprised of various subsystems which are designed by different manufacturers. Increase in number of technology solution providers has led to

provision of innovative automated processes on the industrial floor but at the same time, it has increased the system complexity. Therefore, it has become difficult to integrate various processes on the production floor.

Difficulty in system integration arises from the inter dependencies of subsystems, and the communication requirements between them. This thesis is focused on providing a solution to reduce system integration complexity and aims at providing a way to ensure process uniformity. It is important to define the common interfaces to reduce system integration complications.

Furthermore, during application development process, the subsystems that are implemented together are quite dissimilar from each other as these can be built on different logics. Hence, it is also important to test the integration of different subsystems. There can be different data structures for varying modules, and it is necessary to verify the communication between hardware components in this scenario. Automated integration test techniques are usually developed for software systems, where the response is available in milliseconds. Physical systems take more time to generate response since a physical task must be performed before a response is generated.

1.3 Scope

This thesis is aimed at providing a solution to reduce the complexity during the integration phase thus ensuring process uniformity. In addition to that, it aims at providing a solution to use existing software integration test techniques to be implemented for physical system integration testing. Focus of this thesis revolves around these questions:

- How can integration tests be used to reduce complexity in the commissioning of industrial manipulators?
- How can testing used in traditional IT be adapted for testing in physical industrial equipment?
- How can integration tests be automated for the realm of industrial shop floor?

The scope of this thesis is also extended to the implementation of free shape path planning for industrial manipulator.

- How can free shape algorithms be implemented in industrial manipulators?

1.4 Thesis Structure

This thesis consists of five sections, Section one introduces problem statement and provides scope and objective of this thesis. Chapter 2 provides background, existing research and basic concepts related to the problem. Chapter 3 is focused on proposing a methodology to discuss the problem and it also provides the key criteria considered to select the equipment for this project. Chapter 4 delivers the implementation of proposed methodology and highlights the system's functionality flow. Integrated system is verified using integration tests which are explained in chapter 5. Chapter 6 provides a conclusion answering the question proposed in the scope of this thesis.

2. LITERATURE REVIEW

The term “Robotic” is defined as the range of technologies, in the form of physical machines having computational intelligence that gives it ability to perform tasks which cannot be performed using the unintegrated core components alone. [2] Machine’s ability to move on its own make way for a wide range of applications in robotics. Robotic systems are more efficient and precise than humans, faster than machines. Tasks performed by robotic systems are usually complex and cannot be performed by conventional machines. [2]

Robotics have existed since thousands of years. People have been creating robot like machines and devices since Roman-Greece times. But in 1921, a Czech writer Karel Capek introduced the word “Robot” in his science fiction Rossum's Universal Robot. According to Capek, robot is an artificial device without unnecessary qualities of a human such as feelings and it is brilliant at its work. The first human body shaped robot was made by Leonarda da Vinci in 1945. This robot was capable of moving arms and legs.[3]

Cambridge dictionary has defined robot as a machine that is controlled by some computational system such as a computer or completes the given tasks autonomously.[4] While the RIA (Robotics Institute of America) has defined robot as a multifunctional and reprogrammable device that is programmed to perform some certain predefined tasks such as moving an object. While in engineering, robots are defined as multipurpose and complex device that has autonomous control system, a mechanical structure and a sensory system.[5]

In industry, robots are replacing human operators by taking over the tasks that can be harmful or dangerous since 1960s. The introduction of robots in production processes has not only made the production process more accurate and faster but also has opened the doors for researches to develop more intelligent and flexible robots for these processes. Increase in robotic applications and the level of sophistication, in completing a task, introduced by a robot have motivated the research organization to create new needs for robots outside the production processes. Currently, not only the robots are being developed for manufacturing processes, but also there is a vast demand for robots that are service-oriented or can satisfy social need. [6]

In 1920s, a science fiction: The Robot as a Human Servant gave the idea of robot that can serve humans as a servant and thus created the demand for robot servants. Now researchers are aimed at developing service-robots that can fulfil human social needs.

This has changed our market viewpoint from industrial robots to social and personal robots.[6]

According to Gates, the robotic industry today is developing at the same speed the as the computer industry developed three decades ago. Now, on one hand, the industrial robots working in assembly lines are manufacturing automobiles. On the other hand, robots are disposing roadside bombs in Afghanistan and Iraq. They are performing surgeries, cleaning floors. They are also changing our hobbies and toys.[3]

Movies and other science fictions are playing quite a role in popularizing the robots among people. People are becoming more open not only to the idea of robots working and helping in daily lives but also the idea of robot companions. Even though lots of advancement has been made in robotics industry, but we are quite far away from developing those science fiction robots.[3]

Currently, domestic robots are capable for carrying out one dedicated task. These tasks include keeping a diary, message delivery, educational functions, home safety and entertainment. But these robots are not advanced enough to do more than one job with that much accuracy and efficiency. Research is also focused on the interaction of these robots with human. But more and more studies in the field are making it possible for companies to develop multi-tasking robots. Such as, Sony aibo, it can interact with children and can study growth, learning abilities and their emotions.[7] Sony aibo and its design is inspired by dog. Artificial intelligence and advanced electro-mechanical system have made it smart.[8]

According to Asimov, there will be a need to introduce new discipline, for robotic intelligence, under the name “robo-psychology” because of its different from human intelligence. Pransky has introduced the concept of robot assistant as: robotic nanny, robot assistant, robotic butler. Robotic nanny can raise and feed the children; assistant can serve as a secretary at home while butler can help with housework. This seems like a very luxurious future where robots are working as servants. Robotic intelligence will reach to a level where they’ll be evolving and reproducing themselves.[9]

By the year 2006, industrial robot population was 0.95 million out of 4.49 million robot population. It made 21.16% of total population. The qualities of industrial robot defined by RIA are following: Multifunctional, reprogrammable and can move objects. Currently, industrial robots are working in all the manufacturing, packaging, medical, communication, optical and food industries ranging from optical electronics manufacturing to automotive manufacturing.[10]

In Industry, robotic automation has helped with workplace safety, reduction in labour costs, increase in productivity, better product quality and in making the production process more consistent and a faster. Due to Environmental regulations, and global competitiveness, all the manufacturing and production industries to continue research in making their production process better by introducing and improving automation.[10]

Batch production of products has been the focus of industrial automation to this date. Product quality increase in productivity and better life at workplace has been the aims of automation. In japan, MITI has been focused on the concept of fully automated factories by aiming their research at machine vision, machine understanding of spoken languages. Material handling in assembly lines by industrial robots, visual inspection of process and product is also a focus of this research.[11]

Table 1. *IR application Percentages according to 1996 RIA data*

Application	**Percentage shared in IR applications**
Welding Process	41%
Material Handling	27%
Coating and painting	20%
Material Removal, Dispensing and Assembly Applications	4%

Table 1 below shows the percentage in which IR were deployed in different industries according to 1996 RIA data. Table shows that, at that time, robots were more common in welding and material handling. By 2008, not only the IR maintained their position in these applications, but they were also introduced in food and automotive industry.[10]

Robot manufacturers are also focused on multi-robot controls, because robot working in pipeline reduces the production cost and improve the speed of the production process. Since multiple robots are being controlled by one controller, it helps to avoid collision, saves floor space. Multi robot system is made more flexible when robot is used to hold the work price and other robot works on it.[12]

Wireless communication between robot sensors and controller is also under development. Although, this communication is quite efficient already, but wireless communication will help in case of emergency and safety. Wireless communication between teach panel and robot controller will make it safer for user. Machine vision has been in use for force control in robotic applications for a long time now. Future robotic applications look very promising where they will be able to perform tasks like cutting, sorting, cleaning with higher level machine vision. [12]

In short, IR are being used in all most every modern industry to help in improving the manufacturing and production systems, in increasing the efficiency, in helping the globalization, demographic and environmental issues, cost efficiency, in improving the work place environment more human friendly.[10] But to ensure safety in work place environment, there are three laws of robotics:

1. A robot must not injure or hurt a human being.
2. A robot must obey the orders given by human being if it doesn't conflict with first law.
3. A robot must protect itself as long as it obeys the both laws given before.[13]

Some books also state a zeroth law that is: A robot must not injure humanity, or let harm humanity through inaction.[14]

For safety reasons, IR were placed in a cage. But now, they have started to come out of their cages. Input/output systems and fail-safe buses and real time robot controller are helping with safety issues. EsaLAN Systems developed in 2006 is a development in robotic workplace safety to make sure that those laws are followed. It has introduced the software limits to limit the working range of a robot.[12]

Table 2. *Facts: Robotics Today [5]*

Organizations in the Field of Robotics	More than one thousand
Magazines about robotic	More than five hundred
Yearly conferences about robotics	One hundred or more
Degrees in robotics	Fifty or more

Table 2 states the facts given by Jazar in his book: Theory of Applied Robotics. According to Jazar, robots are better than humans at completing a task with higher accuracy and precision. Moreover, robots do not have human like needs of fresh air, suitable temperature and proper lightening. They can work in any kind of environment. That is the reason, robots are becoming more common in industries and a large number of industrial applications depend on them.[5]

2.1 Background and History

In history, robots are highlighted by science fiction and cinema writers. Those writers gave birth to a fantasy that has been changed to reality. Since, in fiction, robots are usually given a human like form, so, the definitions of a robot given by different experts varies from a multipurpose, reprogrammable industrial manipulator to a humanoid machine that is able to perform tasks like a human.[14]

The word robot was introduced in last century, but the field or robotics has existed since long before that. As described earlier the word robot was introduced by Capek in his play

fiction Rossum's Universal Robots which was published in 1923. R.U.R was premiered in1921 for first time. Hence the word robot dates to 1921. The word robot seems to be derived from two Czech word robota and robotnik which means servitude and peasant respectively. Before Capek, word "automation" was in use instead of "robot".

The word Robotics was first used by another science fiction writer Isaac Asimov. In his book "*I, Robot*", which was published in 1950, Asimov wrote robot-themed short stories. In these short stories, he stated the "Three Laws of Robotics". These laws have been stated above in this document. Later, 1985, he introduces 4th law which was called "Zeroth Law". So, The world of robotics have Capek and Asimov to thank for their contribution to the field in the form science fictions.[14]

2.1.1 **Industrial Robots**

The first industrial robot that made it to market was The Unimate #001. In 1956, George C. Devol and Engelberger met at a cocktail party and started talking about the Asimov's science fiction which started a partnership between these two, which led to this invention. Engelberger is known as "Father of Robotics". The first robot was installed at General Motors' assembly line in 1959 and it was controlled by cams and limit switches. The first mass produced robots were Unimate 1900. By 1961, around 450 robots were being used in die casting. [14], [15]

In 1960, another robot Versatran (derived from words versatile transfer) mean was developed by two brilliant minds at AMF: Veljko Milenkovic and Harry Johnson. Later, six Versatrans were installed at Ford factory in Canton and by 1963, robot was commercially available.

A Norwegian company Trallfa Nills Underhaug designed a hydraulic robot named Trallfa robot in 1964. The robots were designed due to the shortage of labour in wheelbarrow manufacturing. The robot had five or sex DOF and were used for wheelbarrows' painting purposes. Continuous path motion and revolute coordinate systems were used for the first ever time in these robots. Later in 1976, these robots were modified by Sims, Jefferies and Ransome for arc welding application. Machine vision was first introduced by GM in installing the Consight system in 1970. [14]

JIRA (Japanese Industrial Robot Association) was founded in 1971, because of the fact the Japan was making vast advancement in using robots in manufacturing. In 1974, Kawasaki improved the Unimate robot's design to use in arc welding of motorcycle frames. A Unimation robot assembly line was already in work at their Nisan plant since 1972.

Meanwhile in 1973, for their Hi-T-Hand robot, Hitachi created force controlling and touch sensing abilities.[14]

First micro-computer controlled, Hydraulic actuated Industrial robot T3 (The Tomorrow Tool) was developed by Cincinnati Milacron Corporation in 1973. The robot was used for transferring bumpers, welding and loading machine tools in automobile industry. The robot was later upgraded to perform drilling tasks in 1975.[14]

Victor Scheinman, in 1974, designed a robotic arm that was controlled by a minicomputer. This arm was named as Vicarm but renowned as standard arm. PUMA was developed from Scheinman's Vicarm by GM in 1977.[14]

In 1970, a Swedish company ASEA Group of Vaster created automated electric IRb-6 and IRb-60 robot for grinding tasks. Later in 1977, ASEA created two more microcomputer-controlled robot that were powered by electricity. 1988, Brown Boveri Ltd and ASEA merged to form ABB. ABB is working in automation and power technology and it is one of the leading companies today.[14]

In 1979, SCARA, a robot with revolute joints, was designed by the joint efforts of IBM, Yamanash University and Sankyo. Since joints were revolute, its arm was rigid and provided better assembly for vertical tasks. In 1983, company under the name Adept Technology was founded and it introduced its first SCARA robot under the name AdeptOne SACARA. In 2015, Adept Technology merged with OMRON. OMRON adept is now providing robots in all the services industries from automotive, electronics to research labs.[14]

2.1.2 **Non-Industrial Robots**

Mariner 2 was the first space probe that was used in space exploration. In 1962, it was passed as close as 34,400 kms from Venus, and it gathered data about temperature, and atmosphere and sent it back to Earth. Venera 7 was the first spacecraft to land on another planet. It landed on Venus in 1970 and transmitted the data back to Earth, though transmission was limited because of very high temperature on the planet. In 1982, another soviet lander, Venera 7 landed on Venus and sent coloured pictured from the there. Another achievement of Venera 7 was that it took surface samples by drilling, analysed the data, and transmitted the result back to Earth.[14]

Mariner 10 was first space probe that was sent to two planets: Venus and Mercury. Mariner 10 was first sent to Venus and it used Venus' gravitational pull to enter Mercury's orbit. Between 1974 and 1975, it crossed Mercury thrice at 203 kms. It took 2800 pictures and transmitted them back to Earth.[14]

In 1975, NASA sent two probes Viking 1 and Viking 2 to Mars. Soon after the spacecrafts started orbiting the planet, both landers were sent the surface of Mars leaving the orbiters in space. Both orbiters kept transmitting the photographs and results of biological experiments back to Earth for an extended period of time: till 1980 and 1978 respectively.[14]

The Opportunity rover, one of NASA's twin rovers: Spirit and Opportunity. They were launched in 2003 and landed in 2004 on two different places on Mars. There rovers were provided with microscope imager, panoramic camera and spectrometer. Their mission was to find water and analyse surface and capture images.[14]

Falcon 9, rocket launched by SpaceX in 2017, launched a geosynchronous satellite. This rocket was first launched in 2016 to supply cargo at NASA's space station. The relaunch of the rocket in 2017 shows that a milestone to reuse a rocket has been achieved.[16]

Similar to space programs, robots also find their application in military and police tasks from sweeping landmines to sending a traffic stop robot to print speeding ticket on highway.[17] Developing a robot that can identify landmines just like human's is a topic of interest for humans that can help save lives of many soldiers. Icosystems, swarm intelligence-based robots, a system of 120 robots. These robots are will use tail and fail methodology to search for landmines or most efficient paths rescue paths.[14]

Military drones are quite popular among U.S. military since they can use them to attack terrorists from the air. First drone Predator UAV was used in 2002 to destroy Al-Qaeda militants in Afghanistan, though drones had been under military use since 1999 for investigation purposes. DARPA funded researchers are also focused on designing robots that can monitor the secure areas in remote locations and can inform military base in case of a break in.[14]

Apart from saving lives in military operations, robots have been in use for medical application since past 20 years. They find applications from developing medicines to performing surgeries. In 1984, Engelberger developed HelpMate robot. In 1988, these robots were helping in delivering medical supplies at hospital wards.[14]

High precision and absence of human like feelings like trembling and shaking makes robots more useful in performing surgeries than humans themselves. In 1990, a device Robodoc was used to hip replacement surgery on a dog, the device was developed by Howard Paul and an orthopaedist doctor Willian Bargar in 1990. Later in 1993, this robot was used in first human surgery. Before Robodoc, surgeons needed to cement the hip to femur by digging down a channel. Over time of 10 to 15 years, cement breaks down and surgery was needed again. Robodoc helped the surgeons to dig channel so precisely that no cement was needed to keep the new substitute hip in place.[14], [18]

Surgical incision effects recovery time of tissues and it may leave a mark. But robotics incisions are small, and which reduces the recovery time a lot and these MIS (minimally invasive surgeries) are quite popular. Because of their popularity they are being used in Endoscopy since 1980s: inserting a tiny sized camera into body through a small incision. This camera helps with surgeries allowing the doctor to see where surgical tool is going.[14]

New robotics systems let a doctor to operate surgeries using a fully remote-control system: jogging camera and tools through a use panel. Even the heart surgery is performed using this remote surgical system.[14]

Development of prosthetics is a quite success in medical robotics. A robotics device replaces the missing body part while providing the natural movements of missing part. In 1988, the first robotic arm, equipped with artificial skin, moveable fingers, motorized shoulder and rotating wrists, was fitted to Mr. Campbell Aird.[14]

Apart from finding applications in industry, space exploration, military and medicine, robots are being used in in other fields as well, such as entrainment industry. In 1988, a robotic toy Furby became popular in market, because it was equipped with sensors to help it react to environment. In addition to its own Furbish language, it could learn English language through human interaction.[14]

In 1988, another robotic toy Lego MINDSTORMS were released. LEGOS are reconfigurable. They became quite popular in educational programs because they were quite helpful in teaching about robotics sensor and actuators.[14]

Inspired from space exploration robots, deep sea explorers are focused on using robotics to explore deep waters. Odysseys llb was developed for sea exploration while Dante II was designed for volcanic exploration. Robots are also being developed for helping in disaster and emergence situations or to provide fast life support such as drone ambulance with oxygen supply.[14]

Robots have already become an important part of our lives. Now, it is much likely that, with the help of robots, we can industrialize the moon. Today, robot technology has become very advanced, but robotics has not reached its saturation point yet. A lot of research and advancement is still going on in the field especially on the robot control because it controls the robot performance. [12], [19]

These days, quadcopters are quite popular because of their applications from photography to use in shipment. Phenox is a reprogrammable, intelligent and interactive quad-

copter developed by Phenox lab, Japan in 2012. FPGA and MCU makes self-localization, self-stabling and fast image processing possible. It is gesture and voice controlled hence interactive, self-stabling and light weight.[20]

In 2005, an Austrian company Schiebel developed CAMCOPTER S-100, that is an automatic drone, that can complete entire mission on its own. It can be used for military application, supply line monitoring or laser scanning and, because of it is completely automated, it is called Unmanned Air System (UAS). It can take-off and land vertically, without any extra help, can operate in any conditions: day, night, bad weather. Its range is 200km and it is has a GPS system that can be programmed for flight direction.[21]

SmartPal is a cleaning robot developed by a Japanese company Yaskawa Electric Corporation. It can detect and pickup boxes and objects from floor using its hands and arms that has 7 DOF. Robot has IR sensor to detect humans, distance sensors to detect relative position, and a camera mounted on the head to detect the objects to pick. It also has wireless communication system to interact with other SmartPals or peripherals like elevators.[22]

Rollin Justin is a mobile humanoid robot developed by German company German Aerospace Center (DLR) in 2008. It finds its applications in household work or to assist astronauts in space. It has stereo camera and motion detection sensors, that only make it capable to reconstruct its environment but also make it possible to move independently and autonomously while avoiding obstacles. It has multiple DOF, can multitask and capable of catching things thrown at it with 80% accuracy. It can serve beverages while monitoring its environment and avoiding obstacles or can even bring coffee from a coffee machine. [23]

In 2005, Boston Dynamics developed a four-legged dog like robot named “BigDog”. It has 16 joints, Gasoline Engine, hydraulic joints and a payload capacity of 45 kg. It has the ability to absorb shocks and can recycle energy while walking. It is equipped with sensors for join force, position, ground load and contact, navigation, stereo camera, and gyroscope. Its local computer based control system monitors and handles user interaction and it can walk on any different kinds of surface while carrying load.[24]

YuMi is first ever collaborative robot developed by ABB in 2015. Since YuMi is a collaborative robot, it brought the idea of humans and robot working side by side, without any cages, to reality. It is intuitive robot that can perform repeated tasks with precision. It is a dual arm robot, with on cameras mounted on gripper and is meant for industrial assembly line operations. Lead-through programming is one of the features of YuMi, that

means, to make this robot able to perform a task, it is not necessary to write code instructions, it can be taught by guiding through the task.[25]

Pi4 Workerbot is a two-armed, humanoid robot developed by Fraunhofer IPK Berlin Germany. It has payload capacity of 10kg per arm and can detect the location of parts by itself. It can load and unload its workstation on its own thus provides completely automated solution. It is safe to work alongside humans and has an integrated safety and force monitoring technology. It has 7 DOF, a forehead mounted 3D camera, two cameras on the sides and an LCD to display smile while working and providing feedback. Impedance control makes it able to adjust to the disturbances and errors. Similar to YuMi, this robot can also be taught by physically guiding through a task making the programming much easier.[33]

2.2 Robot Classification

Robots can be classified based on multiple parameters such as application or based on their movement mechanism: kinematics and locomotion. In case classification is based on later criteria, a robot can be further classified to provide more detailed information about its structure.

2.2.1 Classification based on application [27], [28]

- Aerospace Robots: Aerospace robots are classified from space robots to simple that can fly in the air such as drones.
- Consumer Robots: These are robots that individual can buy and use them at home for fun or to help with simple tasks such as cleaning.
- Disaster Response: These robots are used in case of a disaster as clear from their name. They can be either used to search for survivors or can be used to do aftermaths of a disaster.
- Education Robots: These robots are aimed at helping with education either in classroom or at home such as Legos.
- Entertainment Robots: As clear from their name, these robots are meant for entertainment purposes from making us laugh to playing music.
- Exoskeletons: These robots are used to help with rehabilitation of a physically disabled person such as to help paralyzed patient walk again.
- Humanoid Robots: These are the robots that we usually see in the movies: a robot that looks like a human.
- Industrial Robots: Industrial robots are used to help in industry to perform dangerous or repetitive tasks. These are usually manipulator arms.
- Medical Robots: Medical Robots ranges from the robots that perform surgeries to the first aid robots or robots that lift the equipment.

- Military Robots: These robots help in military operation from bomb disposal to transportation robots. Search and rescue robots are also included in this category.
- Telepresence Robots: These are remote control robots that can help a person walk around without physically being present in that area. A person can login to a robot avatar and can control it from distant place.

2.2.2 Serial vs Parallel Robots

- Serial Robots: These robots have sequentially fashioned links connected via joints, have a fixed base and an end-effector. Example of these robots is a simple industrial manipulator.[29]
- Parallel Robots: They can have prismatic or revolute joints and are shaped like one or more loops. There is no defined first or last link. They workspace of these robots is restricted but they can handle more payload with greater accuracy. [30]

2.2.3 Stationary Robots

Stationary robots are usually serial robots with one end fixed and the other end is open: end-effector. It is used to perform task with the movement of its arms which are called links as stated before. Main types of stationary robots are Cartesian, Cylindrical, Spherical, SCARA, Articulated and Parallel robots as shown in Figure 1.

- Cartesian Robots: These robots have three joints, all of which are prismatic, and they use the cartesian coordinate system i.e. their motion is along x, y and z.[31]
- Cylindrical Robots: These robots have one prismatic and one revolute joint. Revolute joint is usually at the base, so, it has rotational motion on the base and cartesian on the top.[31]
- Spherical Robots: In these robots, while arm is connected to the base with a twisting joint and they have two revolute joints and one prismatic joint. The axes form a polar coordinate system and because of that, these robots are also called Polar robots.[31]
- SCARA Robots: SCARA robot has 2 revolute joints which have parallel axes of rotation and one linear joint, it is compliant to x and y axes, while it is rigid in z axes. It is quite useful for vertical assembly tasks.[31]
- Articulated Robots: All the joints in these robots are revolute joint and they provide a human arm like motion. Number of joints can vary from 2 to 10 or more. [31], [32]
- Parallel Robots: They can have prismatic or revolute joints and are shaped like one or more loops. There is no defined first or last link. They workspace of these robots is restricted but they can handle more payload with greater accuracy. [30]

2.2.4 **Mobile Robots**

Mobile robots are capable of movement i.e. locomotion. Usually, their bases are platforms that make them capable of motion. It can be wheels, legs, drones or swimming robots i.e. robots are capable of locomotion in air, on ground and even under water i.e. they can move around within their predefined workspace.[31]

2.2.5 **Swarm Robots**

Swarm robots is the idea to use multiple robots work in collaboration on a task i.e. team of robots working on a task. They can be mobile robots or manipulators that have a communication and sensor network. Swarm robotics approach is inspired by insects: large grouped robots should be able to coordinate like insects, and to create an intelligent system, individual robots should be able to interact. [33]

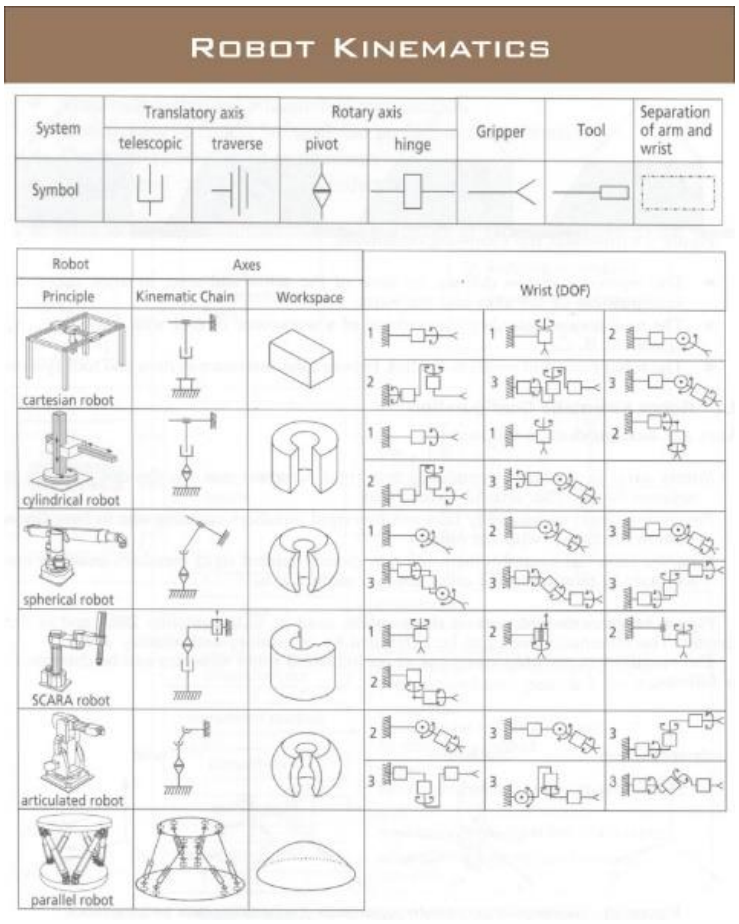


Figure 5. *Stationary Robot Kinematics [34]*

2.2.6 **Classification based on Power Source**

Robots can be classified based on power source. Currently, power sources can be divided into 3 categories: electromotive, pneumatic and hydraulic.[34]

- Hydraulic actuated robots are used for heavy loads, where high power to size ratio is required.
- Pneumatic actuated robots are usually open loop and inexpensive and are used for fast operations.
- Electric robots use electricity to power electric motors: stepper and servo motors. They are usually used for small payloads.

2.2.7 JIRA Classification

JIRA (Japanese Industrial Robot Association) has divided robot in 6 classes naming them from class 1 to class 6.[5]

- Class 1: A device that is manually operated by a user but has multiple DOF.
- Class 2: A device that is preprogramed and performs all the tasks based on that predefined program and this program cannot be changed i.e. fixed sequence robot.
- Class 3: A device that performs tasks according to predetermined program and this program can be changed i.e. programmable device.
- Class 4: Robot/device can be programmed by physically walking through a system. An operator manually operates the robot for first time and robot learns those steps and becomes capable of performing that task on its own.
- Class 5: Instead of manually teaching the robot, operator provides the motion program.
- Class 6: A robot that can learn about changes in its environment and can perform a task successfully while understanding the new environment.

2.3 Assembly Lines

Assembly line is a pipelines system, where a workpiece is passed through different workstations through a transport system like a conveyor belt. Workstations are productive units where different operations are performed on the workpiece to develop a product from it.[35] Assembly lines were invented to make the factory process fast and production cost effective. In assembly line, workpiece passes through stationary workstations, which are ordered in some certain order. Workstation takes some time to process a workpiece, this time is called cycle time, and it is quite optimal to keep the cycle time even for all workstations for smooth flow of operations. This optimal distribution problem is termed as assembly line balancing problem. [36]

In 1798, because of war threat with France, U.S. needed a large quantity of weapons. Problem was solved by Eli Whitney, who distributed the task in workstations by creating templates for each part and then put machines in productions system. Before Eli, a craftsman was expected to be an expert of the whole process, but Eli changed that con-

cept by creating the part templates. In the next century, Elihu Root introduced the concept of: “divide the work and multiply the output”. In 1849, Root divided the operations into very basic units. Even though assembly lines were being used in production before that, but by dividing the tasks into basic operations, Root made the process faster and accurate.[37]

Each worker in assembly line has an optimal working pace. Inaccuracy increases if worked is pushed to work faster than that. According to Fredrick W. Taylor: If work and tools needed for that work are placed in the right order, a worker’s time can be saved. Thus, manufacturing technology owes Taylor for introducing methods of motion and time study. Later, Henry Ford described the similar principle for modern manufacturing process that all the tools and operators in the assembly line should be places in an order to make sure that a workpiece has to travel least possible distance before reaching the finish line.[37]

Ford’s assembly line technique was first used to develop a flywheel magneto. The production time was effectively reduced from 20 minutes to 5 minutes by dividing the task into workstations and placing the workstations at an ideal height and in optimal fashion. Ford also came up with the idea of setting up a pilot plant: Mass production was made more accurate by advance correction of errors in development of process by using the same tools, devices and labour in an assembly line for development of a sample product. All assembly line production systems are now following this standard.[37]

2.4 Assembly Line Methods

Over the years, assembly lines have seen lot many methodologies and production systems. There are various factors that affect these production systems ranging from capital limitations and international, environmental or cultural laws. Some of the popular assembly line production methods are discussed in this section.

2.4.1 Classic Assembly

Classic assembly or team assembly line is manual assembly line where human workers are working on workstations. Work is divided among several workstations where a worker who is expert at one part of the whole process is doing his job on the relevant workstation. The product can be simple, large or complex, but they are identical. The tasks are repetitive and cycle time depends on individual’s working speed and experience.[37]

2.4.2 Automated Assembly

Automated or cell manufacturing assembly is the use of machines or robots in assembly line for the development of robots. Like classic, work is divided among different work-stations and end products are identical but instead of humans, dedicated machines are completing the tasks at hand. Automation can range from user operated machines to fully automated factories. In this assembly, skills are easy to inherit, and system is more accurate and cost effective in long-terms.[38]

2.4.3 Modular Assembly

In Modular Assembly line, the parts are created in subassembly lines separately and then they are fed into the main assembly line for integration. Like in automobile industry, body, interior etc are designed separately and then they are combined. This method reduces the production time since several parts are created in a parallel.[39]

2.4.4 U-shaped Assembly

In U-shaped assembly, workers are placed on the inside of the shape, thus making it easier to communicate and observe the process. This assembly makes it possible to revisit some stations, hence there is no need to duplicate a station. But U-shaped assembly is not as flexible as line assembly since line assemblies can be used in production of multiple design products as the same time. But U-shape is more efficient than line assembly.[36]

2.5 Path Planning

Kinematics is the science of motion which deals with position, velocity and acceleration of body: in kinematics, motion of a body is studied without paying any interest to the force that causes the motion. It involves the study of links: how the move with respect to each other. Kinematics can be divided in 2 categories: forward and inverse kinematics.[40]

In forward kinematics, robot's joint variables are given, which are used to calculate the orientation and position of end-effector. In inverse kinematics, end-effector's orientation and position are known, and joint variables are calculated.

While dynamics is the study of relationship between force/torque and resulting motion. Like kinematics, dynamics can also be divided into 2 categories: forward and inverse kinematics. [5]

- In forward dynamics, joint torque is known and resulting motion/acceleration is calculated from it.

- In inverse dynamics, joint torque is calculated from given acceleration i.e. here acceleration is known.

Kinematics and dynamics are related to control science, which optimizes the system behaviour. Path is geometric description of motion or it can be stated as: set of points that manipulator passes through in order to reach from initial to final position. If time profile is specified for a path, it is called trajectory. Path planning is also a part of control science and it involves: specifying the curve between two points for end-effector to follow, specifying motion between two end-effector positions, time function for motion between initial and final point.[5]

When robot manipulator moves from initial to final point: point to point motion, trajectory generated by the algorithm can be described by a cubic polynomial given by equation (1).

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \quad (1)$$
$$q'(t) = 3a_3t^2 + 2a_2t + a_1 \quad (2)$$
$$q''(t) = 6a_3t + 2a_2 \quad (3)$$

Equation (1), (2) and (3) represents position, velocity and acceleration respectively. In motion from initial point to final point, boundary conditions can be written as given below.

$$q(0) = q_0 \qquad q'(0) = 0 \qquad q(t_f) = q_f \qquad q'(t_f) = 0$$

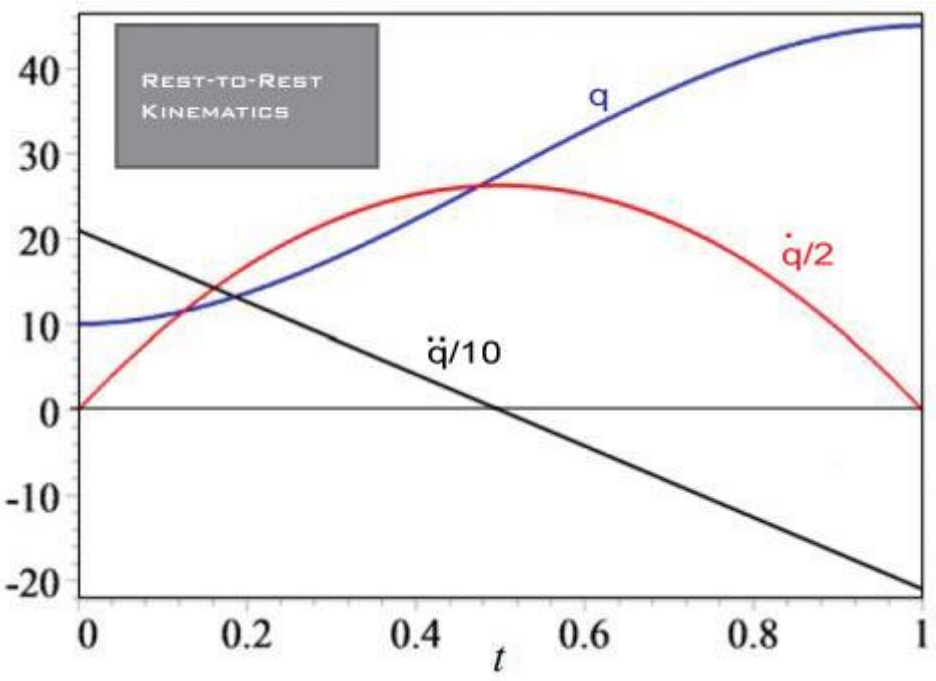


Figure 6. *Point-to-Point Motion Kinematic [5]*

2.5.1 **Algorithms**

Drawing is an important activity and a part of almost every human being. Now, it is becoming a part of robot's life also. Different algorithms, techniques and robotic systems have been developed for drawing and or mobile robot's trajectory planning. Some of these systems and algorithms are discussed in this section.

In tasks like welding, drawing or walking, path followed by end-effector is important and complex. For such tasks, n-points must be defined, other than initial and final points, for end-effector to follow. These points are called via points and method to follow the path from first to last point is needed.[41]

Moving a robot from one point to other, while avoiding the obstacles and keeping the distance as short as possible is a challenge for path planning. In BFS, a robot while at root node (initial point), starts planning all the possible paths in all probable directions i.e. robot starts planning next nodes. While planning the next nodes, the possibility of going outside the workspace, nodes that already have been visited and nodes with obstacles are ruled out. If successor node is not the final node, then BFS keep expanding the next node till it reaches the destination point.[42]

BFS is like a tree, with FIFO array that is populated with nodes. FIFO means that nodes are executed in the order they were added in the list, and oldest node in the array will be executed next. FIFO array keeps adding possible nodes till the final point is reached. When Final point is reached, Robot can map a path of nodes that lead to that point.[42]

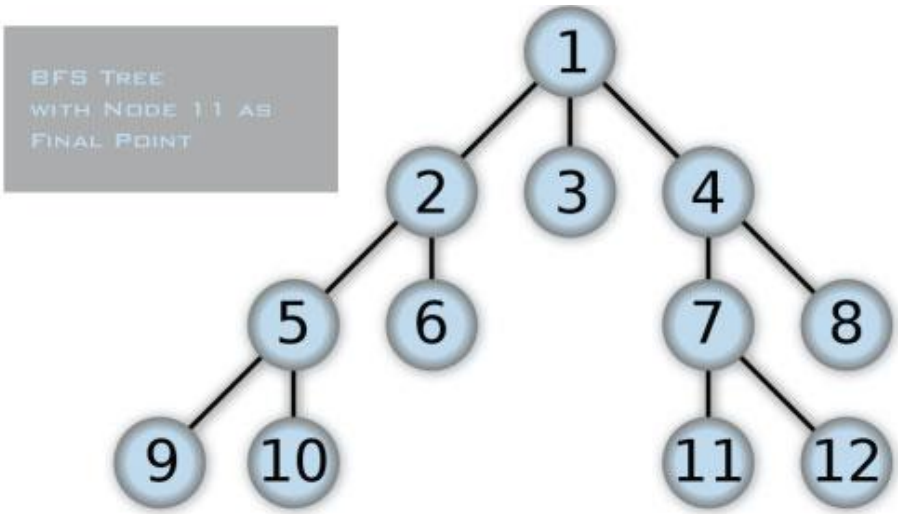


Figure 7. *BFS Tree [43]*

BFS will keep populating the tree starting from initial point which is 1 in this case. The next possible nodes are 2,3 and 4. Since it is FIFO, BFS will execute 2,3 and 5 next and will keep on populating the oldest present node first, till it reaches 11, which is final node.

If solution exists, it can always be found with BFS. Hence, in robotics, it can be used to find the path from initial to final point. But BFS can only be used for a small workplace, because it stores every node in memory. For large workspaces, the number of nodes that should be stored in memory grows exponentially and hence a large memory is needed.[42]

DFS is like BFS in a sense that algorithm starts from initial point and keep on tracing till it reaches the final point. But contrary to BFS, DFS is LIFO. The last added node is executed first. That means DFS will find all possible nodes for the newest added node and since it is LIFO, the next node that will be executed, is the last added node in the array. DFS keeps on expanding till it reaches a dead end, after that it starts expanding the node that was added last but has not been expended yet.[42]

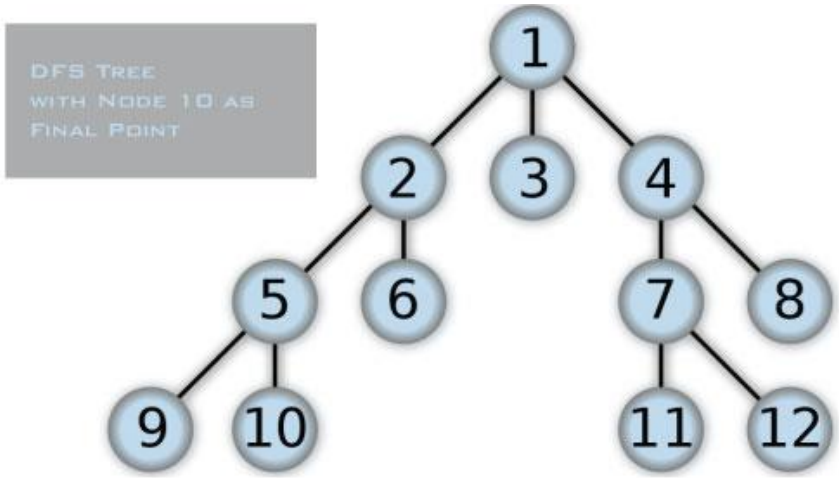


Figure 8. *DFS Tree [60]*

DFS will keep populating the tree, starting from 1, and it will add 2,3 and 4. Now, it will expand last added node, which is 4, and it will keep expanding newest added nodes till it reaches 11. After 11, it will move to 3 and then 2, since they were last added nodes in the stack in that order. It will keep on doing so, till it reaches 10, which is final point here.

Since DFS does not expand and store every node in the memory, it uses less memory than DFS. But problem with DFS is that it keeps expanding newest added nodes, till it reaches the final point. As soon as DFS reaches final point, it stops expanding and returns that path, but that path is not necessarily the shortest path. Other problem with

DFS is that it keeps on expanding newest node forever without any certainty that required end point exists on that path or not.[42]

BFS calculates the path that has smallest number of nodes to reach the final point, but it does not have any information on the distance between the nodes or how far the robot has travelled. Hence, Dijkstra provides a solution for that. Dijkstra is a special case of BFS, which provides information about the distance between 2 nodes: it executes nodes in increasing order of their distance from the root node and selects the node with shortest distance. Hence, Dijkstra makes a path of nodes by selecting the nodes with shortest distances from the root nodes, and hence comes up with path of shortest possible distance.[44]

In some system, visual feedback is being used to minimize the error, these systems are called visual servoing control systems. Usually the error in a conventional system is caused by called math processing error: error in current or upcoming state of the system. But in visual feedback systems, the error is caused by image that is taken by a camera. These cameras are either attached to end-effector or mounted on some other place from where they can observe the workspace. The visual servoing system offers higher flexibility as compared to the conventional sensor systems. Visual servoing system either use a position-based or image-based technique for feedback, they are called PVBS and IVBS respectively.[45]

Visual servoing control system works by minimizing the error in between the target position and robot's position through a visual feedback system. In PVBS, 3-D target features should be defined, and it is quite sensitive to robot calibration and camera errors. In PVBS, 3-D features of images taken by camera are used to calculate the error between robot position and target.[46]

In IVBS, there is no need to define target features, positioning error can be calculated from the images directly, so feedback control is directly enclosed in the image. But IVBS is stable only around the wanted pose.[46]

Fioravanti et el (2008) has defined a modelling technique for positioning tasks using IVBS, for 6 DOF manipulator with a camera mounted on end-effector, using a fixed target as a reference. To calculate the error between the desired and actual position, the control defined is using camera's geometric projection model. which is Euclidean geometry mapped into digital image. They used Euclidean homography structure for image path planning. [46]

2.5.2 **Free Shape Algorithm**

Bezier curves are used for smooth path planning, such as, in robotics, Bezier curves are used in planning the movement of arm for welding. These curves are used for modelling indefinitely scaled smooth path or curves. A Bezier curve, having k+1 control points, can be represented by the following summation:

$$\mathbf{P}(t) = \sum_{j=0}^{k} B_j^k(\tau)\mathbf{P}_j \qquad (4)$$

Where $B_j^k(\tau)$ can be given by Bernstein polynomial. The curves are useful for path planning because they always start and end at $\mathbf{P}_0$ and $\mathbf{P}_k$ respectively, and at the start and end point they are always tangent to the $\overline{\mathbf{P}_0\mathbf{P}_1}$ and $\overline{\mathbf{P}_{k-1}\mathbf{P}_k}$ respectively.[47]

De Casteljau Algorithm is easier to implement and relatively faster method and it controls the end effector by generating Bezier curve.[48] Casteljau's algorithm divides the Bezier curves into 2 sub Bezier curve segments. The control points in this case can be denoted by the summation:

$$\mathbf{P}_j^k = (1-\tau)\,\mathbf{P}_j^{k-1} + \tau\mathbf{P}_{j+1}^{k-1} \qquad (5)$$

Where k = 1,2,3, . . ., n and j = 0,1,2, . . ., n-k, $\tau \in (0,1)$.

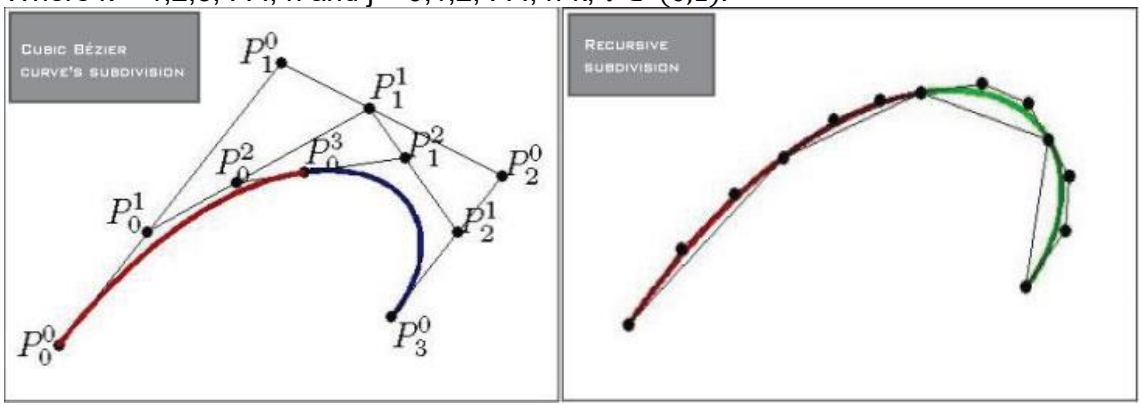


Figure 5. *Cubic Bezier Curves [47]*

2.5.3 **Drawing Bots**

Kotani and Tellex, from Brown university, have developed a robot that can draw and copy handwritings from a given image. This robot can copy ten different languages on paper, white board and can draw characters from language without learning those languages, it just replicates the given bitmap image of characters. They have divided the problem in two different scales: local scale and global scale. Local scale is a window of 5x5 pixels while global scale is whole is the whole image.[49]

To reproduce target image, Kotani and Tellex defined it as binary image of 100x100 pixels. Based on the image, they created an action sequence, that robot follows to reproduce image. The pen/writing brush is shifted in x and y axis by ∆x and ∆y respectively and Boolean variable defines if pen should draw or not, it is represented as (∆x, ∆y, touch).[49]

They divided the task in two scales: local and global scale. Since, local scale is 5x5 pixels, it defines where the pen will move in that limited pixels environment. When the pen has covered those 5x5 pixels, the global model is used to define the starting point of next stroke. These steps are repeated till the whole action sequence has been completed. [49]

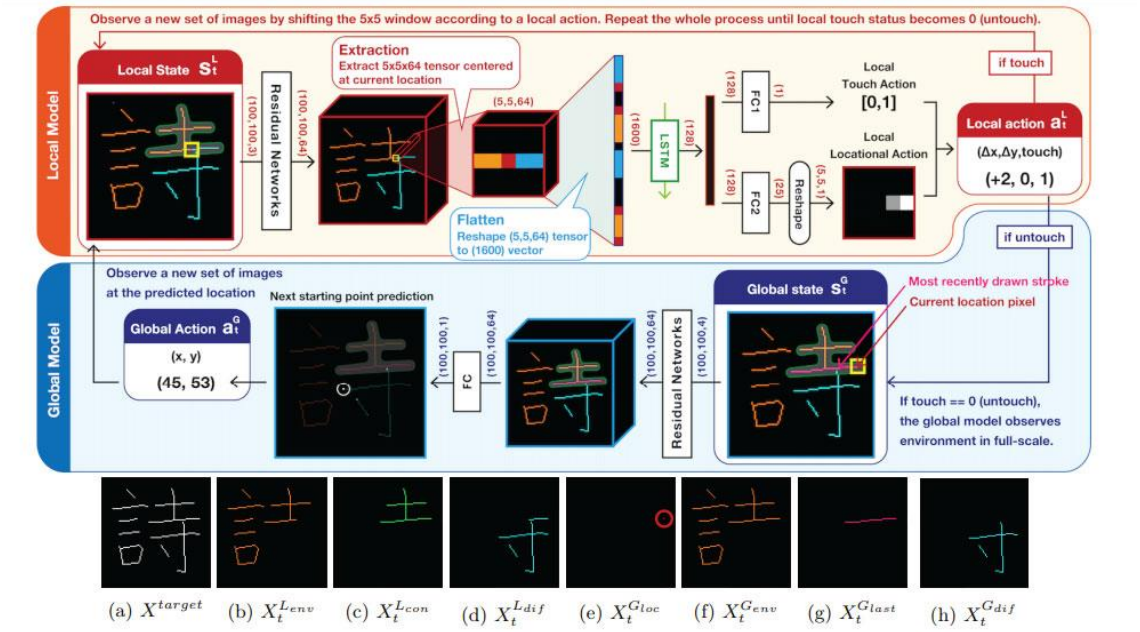


Figure 6. *Network architecture by Kotani and Tellex [49]*

Where X^{target} is the target image that should be drawn, $X_t^{L_{env}}, X_t^{L_{con}}, X_t^{L_{dif}}$ are local model's already visited locations and difference between target and current image i.e. future locations respectively. Rest of the terms are coming from the global scale, the represent current location, already visited location, current local model's recent location, and difference between the current global and yet to visit regions of target image respectively.[49]

A humanoid robot by Lau et el., called betty, for portrait drawings using furthest neighbour theta graphs. For face detection, they have used OpenCV and to compute line-art portraits, they have used Canny edge detection. Line-art portrait can be converted to robot kinematics.[50]

Figure 7 shows the flow chart implementation of the robot. Picture fed to the robot is first converted to a grey-scale image and then, it is converted to a binary image to remove noise by applying a threshold. Threshold has a fixed value, and pixels that do not match the threshold are filtered. Next step is to produce Boolean image from the given filtered image, this is done by using Canny edge detector that returns only black and white pixel image.[50]

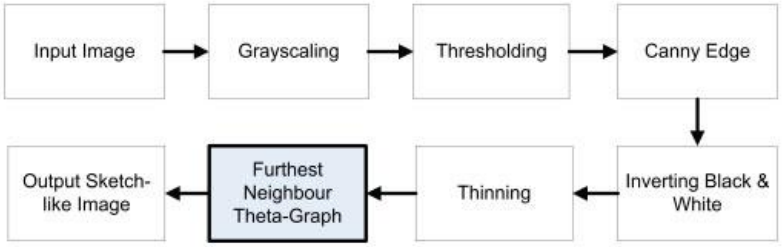


Figure 7. *Flow chart of portrait drawing robot [50]*

Black and white pixels are inverted in next step using XOR operation and then, convolution is used to reduce the number of pixels to an acceptable value. Last step is the use of furthest neighbour theta-graph algorithm for portrait sketching.[50]

Lu, Lam and Yam have proposed a method for drawing by robotic manipulator of 5 DOF, though, they have used only 3 DOF in the described method. They have placed the drawing paper right below the gripper that is holding the pen, while a camera is used for feedback. Since, camera is mounted on 30 degrees with respect to paper, the image obtained by camera is distorted. Lu et el. have proposed a homographic equation to obtain the rectified image from the captured image.[51]

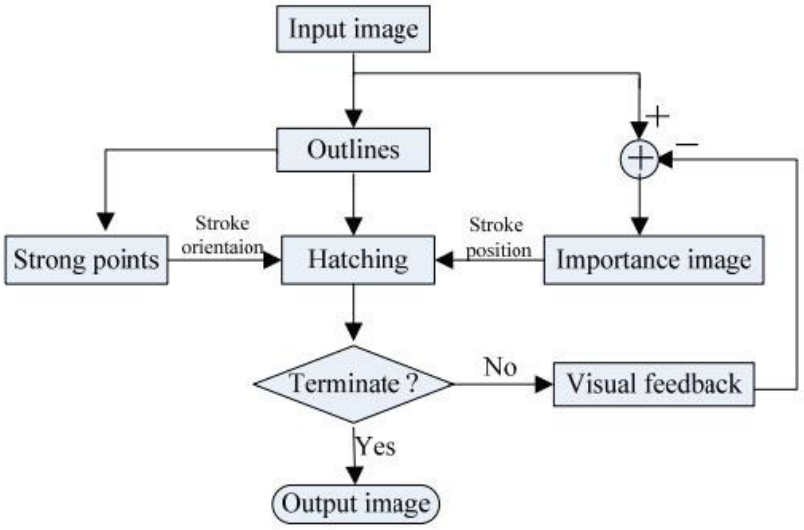


Figure 8. *Flow chart of pen-and-ink drawing robot algorithm [51]*

In this pen-and-ink method, outlines are important, since they express structural information. The outlines of lower structural importance are drawn thinner than those of higher importance. First, image scale space is constructed by convoluting the image by first order derivative of Gaussian kernel. In second step, the Canny edge detector is used to obtain edges, which are then traced. Scale space lifetime is used to measure the structural importance of an edge.[51]

After portraying the outlines, hatching process starts. In hatching, camera feedback is used to describe the stroke positions, while gradient describes its orientation. Iterative hatching process continues until criterion function discovers that structural importance has dropped below threshold, thus criterion function terminates the process.[51]

Calinon, Epiney and Billard's humanoid robot that draws face of person, facing the robot's camera, on a paper. They used Fujitsu HOAP2 robot, an external webcam for human face tracking purposes. Developers used OpenCV to capture and process the face image, while robot's built-in controller library is used for robot movements. As soon as a person sits on the table in front of the robot, robot asks for permission to draw person's avatar. A speaker and mic has been used for this purpose, and if person gives permission, robot starts the process by taking a facial picture of the person.[52]

Facial picture is then converted to a black and white image and a threshold criterion is selected. The selection of threshold is dependent on the mean square error. Calinon et el. have used inverse-kinematic model to define the workspace, and to search for drawing plane. Robot starts drawing according to the Freeman code length, it starts from a rough contour and keeps on adding details. Spline fit of order 3 has been used with 2D trajectory of freeman code to make sure the smooth movement. Meanwhile, to move

from one contour to other, a 3rd vertical dimension is added to this 2D trajectory. After the robot has drawn a picture, it draws a frame around it and signs it using the pre-set trajectory.[52]

2.6 System Testing

2.6.1 Unit Testing

Unit testing is quite necessary part of software or system development. In unit testing, isolated parts or modules of a system are tested. This isolated unit can be a code a procedure or function and developers use unit testing to verify their work. Since, the goal of unit testing is to assure that system is working according to the client's requirements, thus, a thorough system testing is desired, which makes the testing process as expensive as the system development. [53]

Usually, unit test is carried out before integration testing, by executing small process or program that provides the input to unit of the system. The output of the unit is monitored, while trying to maintain the execution environmental conditions close to the actual environment.[53]

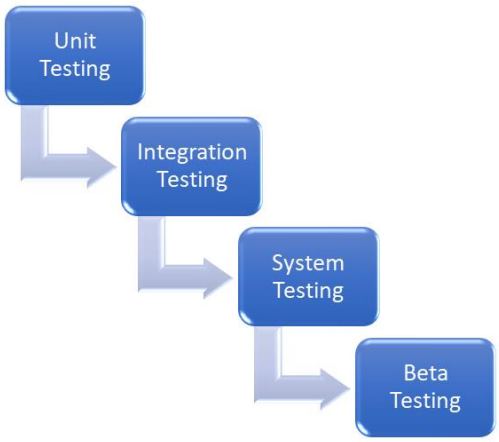


Figure 9. *System Testing Levels*

Unit testing is first stage of system testing, while Beta testing is last. Beta testing is post development testing and it is also known as Application Testing. Unit testing saves time and helps fix the bugs in time. Since, unit testing is performed on small units, it makes it quite easier for developers to reuse small parts of code.[54] Unit testing can be either state based or interaction based testing. State based testing is used to test output of a system to a given input while interaction based testing is used to verify that system can invoke different services when called.[55], [56]

Currently, developers can seek assistance from already present testing tools such as: JUnits and EMMA for Java language[57], [58], Nunit for .net language[59] and PHPUnit is for php programmes[60].

2.6.2 **Integration Testing**

Integration test are carried out on complete integrated systems to compare their behaviour against specified requirements. Integrated system can be software or hardware system. These tests are usually carried out to ensure that system is working according to client's requirements before handing over the system to customer. In case of some hardware systems, sometimes the simulation models are used to test full load scenarios and system failure safety responses. System design, used equipment and performance are usually the main characteristic that are tested in Integration tests.[61]

System Integration Testing (SIT) verify the communication and interaction between the different modules and components of a system. SIT is also used to verify system's behaviour along with other systems. In this case, systems or modules are first verified individually and then they are tested as an integrated system.[61]

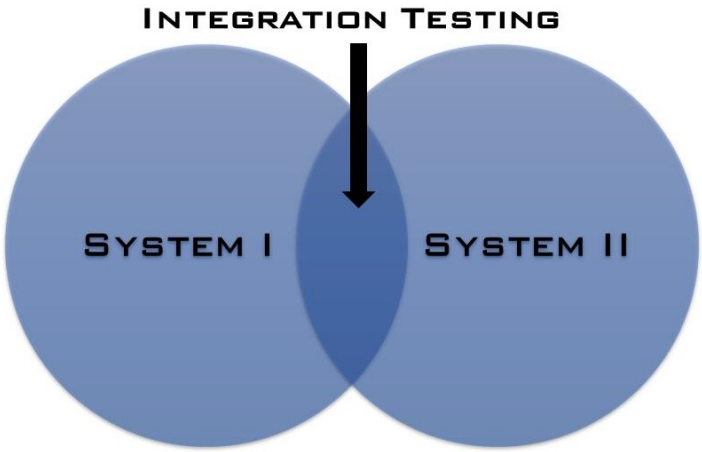


Figure 10. *System Integration Testing*

SIT are performed to verify the system under following conditions[62]

- System is under Thermal Load
- System is working on normal and/or emergency power conditions
- Several modes of system's mechanical operations
- System failure
- Under different temperature, humidity and heat load conditions

Software integration testing is done in host environment, while keeping the target environment condition. Then these testing is repeated in host environment. In case, the software is quite large, then integration is divided into several levels. Lower levels are tested in host environment while higher levels are tested in target environment since, for some systems, system and target environment coupling can be strong that makes is impossible to perform integration test in host environment.[61]

Three levels of system testing are: Intra-system testing, Inter-system testing and Pair-wise testing. Intra-system testing focuses on combining the systems together to make a unified system. This is a lower level of testing. Inter-system testing is focused on building independently tested systems, it is a higher level of integration. Pair-wise testing is used to test two connected modules at a time. It is used to ensure that two modules can work perfectly when combined.[63]

Normally, a system is integrated in advance and then whole system it is tested. But error probability is quite high in this case, several system errors may appear. Identification and correction of these errors in the integrated system is quite difficult, and there is a high probability that after correction of these errors, some new faults can appear. To avoid this situation, several incremental integration testing approaches have been in use.[61]

Incremental Integration testing divides the testing process in different steps. Modules are first tested separately and after that, testing is performed on combination of modules. Since the small chunks of program or small parts of system are being tested, it is easier to detect and remove errors.[61] Three types of incremental testing methods are listed here: Top down integration approach, Bottom up integration approach and functional incremental integration approach.[64]

Top down integration approach, as name suggests, starts from the top and it increments in accordance with depth first or breadth first algorithm, that have been explained in section 2.6. User interface is tested at the beginning of process, and then system keeps on integrating new modules. These new modules are called stubs and they are temporary replacement for the actual modules. Stubs behave in the same way as original module. Top down integration testing determines the confidence level by calculating the threshold failure density of the system. This approach is applied to the systems that have already been tested, and process ends when the desired confidence level has been achieved. In case of failure, system increases the number of test-cases and keeps on testing the decomposed modules till the desired confidence level is achieved.[61], [65]

Unlike the top down testing, bottom up system starts from the bottom. Modules are tested first and then system starts testing combination of integrated modules. Since the modules needed for testing are already present, so, there is no need for stubs. Input and output of integrated modules it defined by drivers. As modules combine, required number of drivers reduces, since now, there are a smaller number of decomposed modules. Drivers assist in simulating the interface with top level modules which are yet to be integrated.[61]

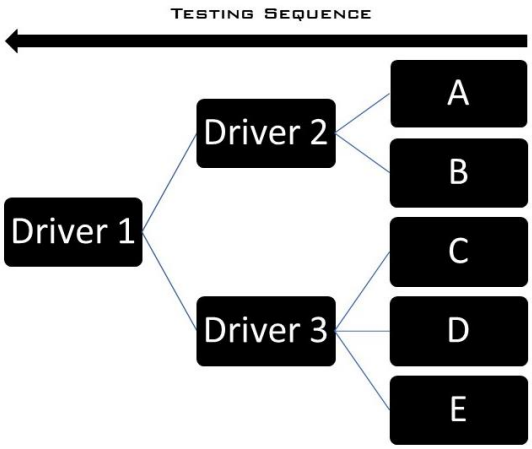


Figure 11. *Bottom Down Integration Testing*

Functional testing is based on system's functional requirements i.e. system is taken as integrated cluster of functions rather than modules. In this approach, system's output is tested against given input, while program's structure is ignored. System is treated like a black box and only the resulted output or response is taken into consideration. [66]

2.6.3 **Regression Testing**

Regression testing is performed to ensure that changes made in a system, do not make any harm. It is carried out when a system is modified. Regression testing begins during the development phase when errors are corrected, system is re-evaluated to establish trust in post changes system. Regression Testing is a big part of maintenance phase, a lot many corrections and updates are made in system over time. After a system has been modified, the original test can no longer be used to test the modified system, hence, new tests are developed for verification.[67]

Figure 12. *Regression Testing Methods*

Regression testing can be carried out using these techniques: Retest All, Regression Test Selection, Prioritization of Test cases. Simplest approach is to execute all the original tests. This approach is called retest approach, and it is quite expensive and time consuming because as the software changes, testing programs also need modification and number of tests needed also increases as system grows. Hence, other Regression testing approaches are used for testing. [68]

In order to save time and resources, it is better to test only modified modules or parts of the system, which is carried out by selecting a subset of test cases that were used to test the changed parts. Third approach to regression testing is quite useful in saving time and efforts. Test cases are periodized on the basis of functionality and most important or frequently used application help prioritize test cases.[68]

2.7 Manufacturing Execution Systems

Manufacturing execution systems also known as MES are developed to fill the gap between planning, manufacturing and control systems used on a factory floor. MES is an information system which monitors the production process on factory floor and makes it effective and efficient. MES tracks and collect real-time data of a production process. MES controls the complete production process: taking orders from customers, evaluating and planning resources and then producing the high-quality goods at low cost. [69]

Currently, manufacturing software can be categorized in the following systems: Enterprise Resources Planning (ERP), Supply Chain Management (SCM), Sales and Service Management (SSM), Product and Process Engineering (P&PE) and Controls, Manufacturing Execution Systems (MES). Each of these systems have different functions but MES overlaps all these categories and it links different types of systems.[69]

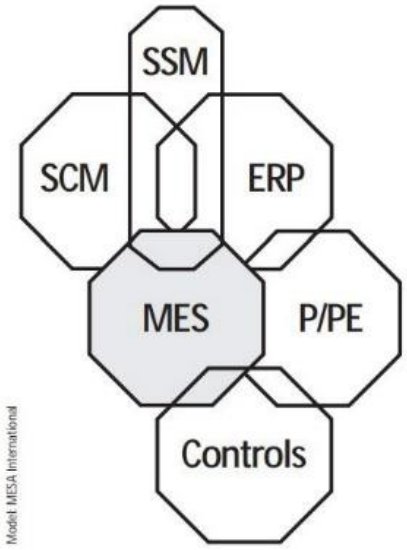


Figure 13. *MES Context Model Concept by MESA International [69]*

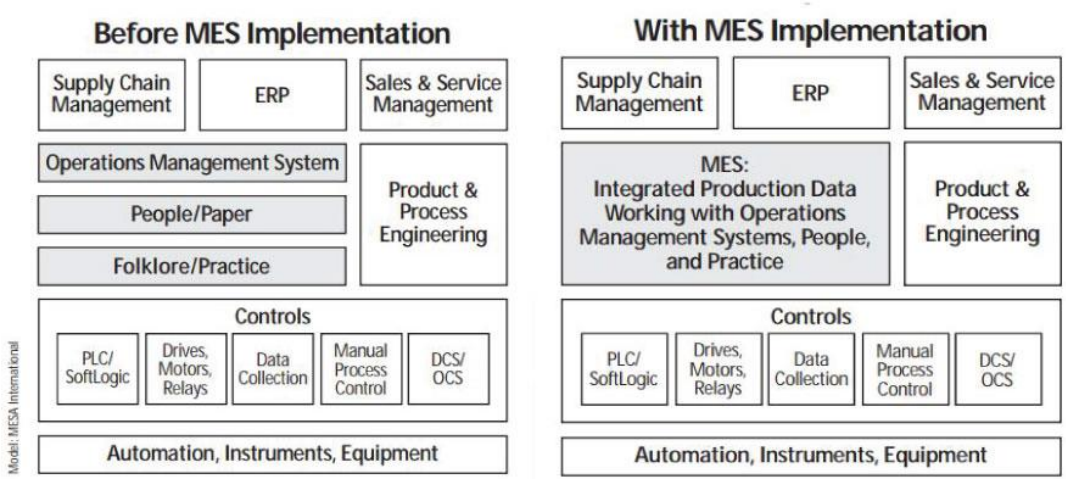


Figure 14. *Plant Information Model by MESA International [69]*

MES provides an intermediate layer between ERP and control systems as shown in figure 14. It integrates planning and manufacturing process in an industry. The lowest level of pyramid of automation consists of physical devices such as sensors and actuators. Layer 1 is called control level and it is made of logical devices such as PLC to perform numerical operations. SCADA is used to access data and provide supervision of control system used in previous layers. Top layer is ERP which is company's management level, this layer corresponds to planning and scheduling of work. While MES is in middles of ERP and SCADA to help and control production scheduling according to the resources and capacity constraints while monitoring the plant operations.[70]

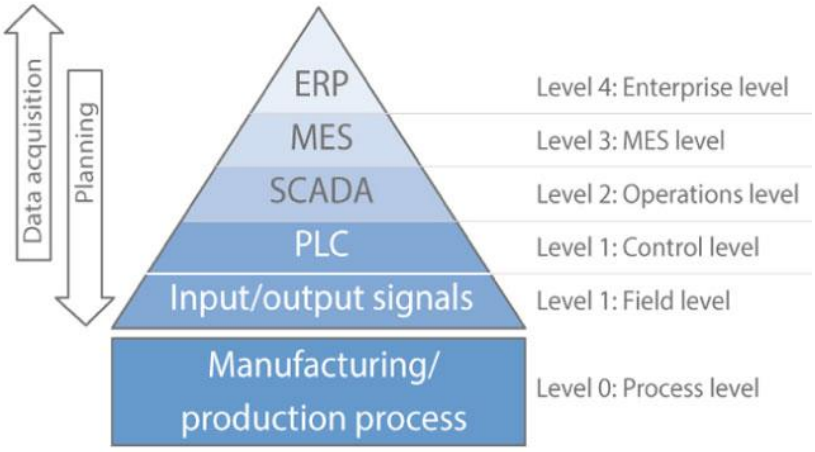


Figure 15. *Pyramid of Automation [71]*

MES provides automation from order request to delivery, from administrative work to technical development and production. According to Kamal, Core of integrated manufacturing is to provide right information at right time and place. MESA provided a model in 1997, defining the functionality of MES. This model is called MES-11 and defined functions are listed in this section.[69]

- Managing and allocating resources in the production plant.
- Scheduling the operations to optimize the production time.
- Dispatching information in order in the production units should work.
- Provides the document that are mandatory for smooth flow of process.
- Collects data and information from factory floor.
- Keeps track and manages labour during the shifts.
- Monitors the product quality and identifies the problem that require attention.
- Keeps track of production and keeps the process flow smooth or assists the operator for correcting in process activities.
- Helps in keeping track of maintenance schedule of equipment to prevent the production failures.
- Provides full history of product during the production process and makes it visible thorough out the process.
- Provide reports and analyse the operation results by comparing them with expected results and history.

2.8 TCP/IP

TCP or Transmission Control Protocol is the basic communication language of the internet. Even though name stands for protocol, but it is in fact a set of protocols, TCP and IP are two main ones. TCP/IP are the set of communication protocols that are used to connect hosts on the internet. TCP/IP defines how data should be divided into packets

and address, how it should be transmitted and received, and how applications can create communication channels over a network.

International Standards Organization (ISO) has defined a model for communication between two systems, regardless of their architecture. OSI model facilitates the communication in between systems without requiring any logical changes in software or hardware. OSI model consists of seven layers. Originally, TCP had 4 layers and since, TCP model was developed before OSI, so, these layers do not exactly correspond.[77]

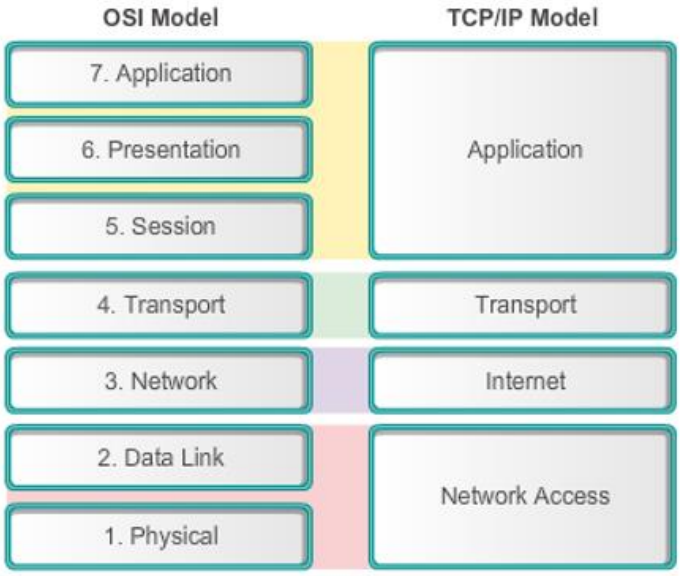


Figure 16. *OSI Model and TCP/IP Model Comparison [82]*

Each layer in TCP model has its only functionality that is based on set of protocols dedicated to that layer:

- Application layer provides application with standardized protocols for data exchange such as HTTP, SMTP, FTP, SNMP and POP3. Application layer uses a port for communication with Transport layer. Ports are numbers and these numbers are associated with protocols.

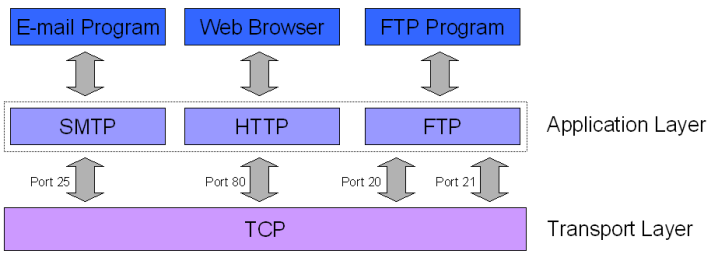


Figure 17. *Application Layer [78]*

- Transport layer (TL) is responsible for host-to-host communication and it is responsible for reliability, multiplexing and flow control

over a network. Transport protocols are TCP or UDP (User Datagram Protocol. UDP is faster than TCP, but it is unreliable.

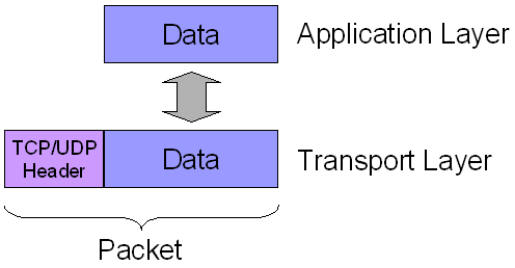


Figure 18. *Transport Layer [78]*

- Internet Layer (IL) connects different networks and it is responsible for transporting data packets across these networks. IL protocols are IP and ICMP (Internet Control message Protocol), ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol). Data packets are sent using IP, while ICMP is used for error reporting.

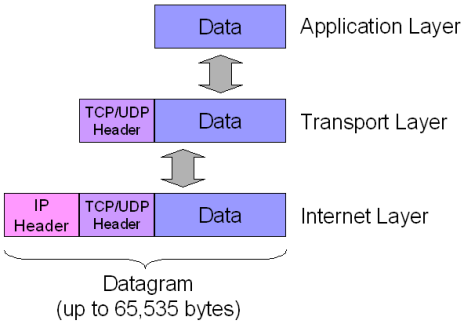


Figure 19. *Internet Layer [78]*

- Data received from TL is divided into Datagrams by IP. Max datagram size can be 65535 bytes. MTU (Maximum Transfer Unit) defines the maximum frame size that can be sent across the network. For Ethernet networks MTU size 1500 bytes.

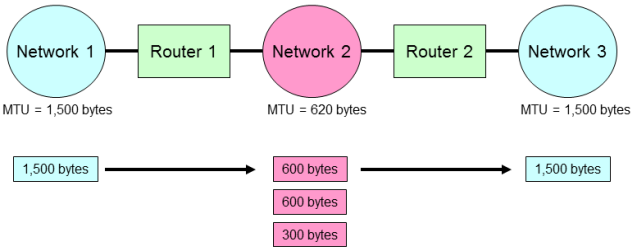


Figure 20. *Datagram Fragmentation [78]*

- Network layer (NL) is defined by physical network that is connected to the computer. This layer defines protocols or methods which operate only on a link. These links interconnect hosts across the network. Ethernet has three layers: LLC (Logic Link Control), MAC (Media Access Control) and Physical.

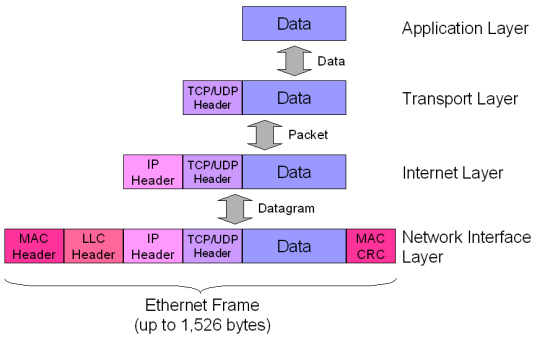


Figure 21. *Network Interface Layer [78]*

In the Updated TCP mode, Network layer is divided into Data link and Physical layer, and Internet layer is changed to Network layer.

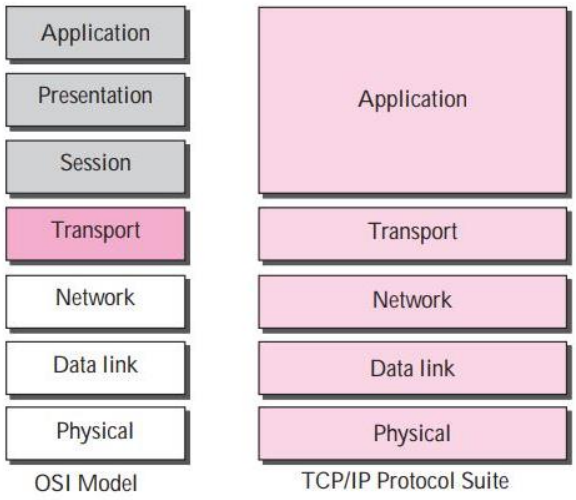


Figure 22. *Updated TCP/IP Model [77]*

2.9 Web Services

MES systems require quite tight connectivity between system, all software should be integrated in way that they can share data and communicate. Different architectures have been under study for this purpose, but Web Services are the most prominent solution in this domain.[72] Web Services are applications that are self-describing and self-contained, WS can be published, located and invoked across the web.[73]

W3C has given an extensive definition of WS, according to them: A WS is software system, which is structured to support interoperable machine-to-machine communication over a network. WS interface is defined in machine-processable format. Other systems use SOAP messages, usually conveyed by using HTTP with an xml serialization or other standards to interact with WS. [74]

According to W3C definition, one main feature of WS is interoperability. WS can be used by any browser or application regardless of platform. Interoperability is aimed at making

and smooth automatic connection from one software or application to other. Reusability is another feature of WS.[73]

Now a days, WS system technologies are leading in the field of information systems, since, WS allows network communication for sharing information and functionalities. In their paper, Qifeng and Zhangjian have proposed a WS-based communication between layers MES system.[75]

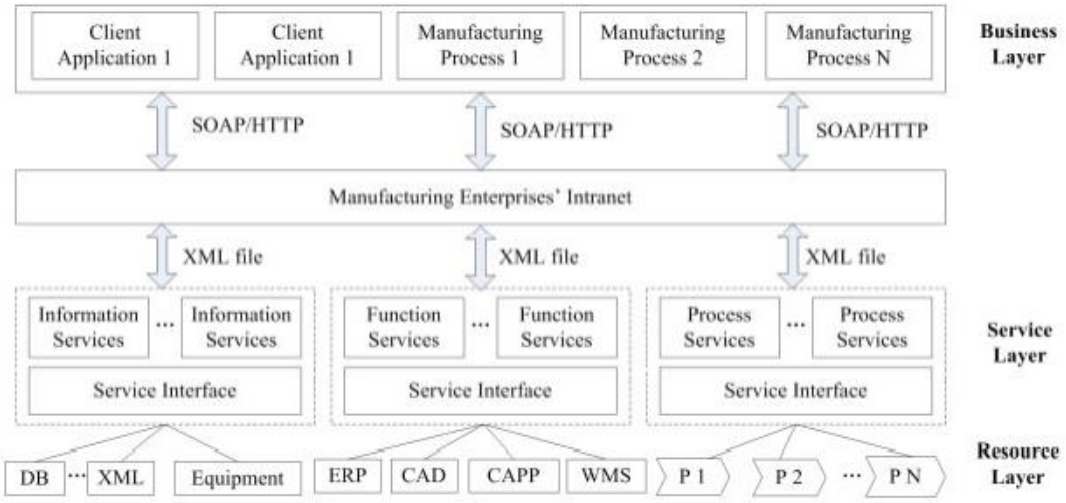


Figure 23. *WS-based MES system integration framework [75]*

The concept of Service-oriented architecture (SOA) has gained quite popularity in MES, because of its flexibility and re-configurability. This high-level service-based architecture provides entirely new perspectives in communication.[76] W3C has defined SOA as a set of components which can which can be invoked, and whose interface description can be published and discovered.[74] SOA is used with in network or services that work together, for providing complete functionality of system.[73]

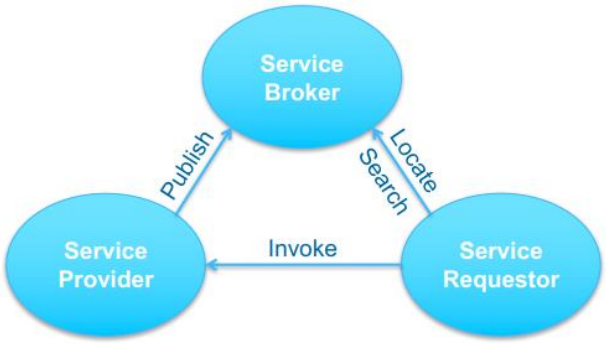


Figure 24. *SOA Components [73]*

Service broker is like a service registry or directory and it is used to locate or publish a service over a network. Interoperability, reusability, message exchange and monitoring, control, transformation and security, and service discovery are the main feature of SOA.

Web represents web-based implementation of SOA, when they establish a communication framework that encapsulates applications within services which interact using a common communication network.[76]

Web services can be used in MES system to control the flow of data, services can be invoked over web by an application client.[72] Web services can be used to monitor the devices on factory floor, while on higher level web services can be used to keep track of work to assist in scheduling and taking new orders from clients.

Web Services are of two kinds: SOAP and REST. Simple Object Access Protocol (SOAP) is a standard communication protocol that is used to exchange XML-based structured information over web services. SOAP uses Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP).[73]

Representational State Transfer (REST) is an architectural style that emphasis on design rules for providing stateless communication and determines the principles for transmitting data. Data is usually transmitted over a standardised interface such as HTTP. In REST, data and services are considered as resources, hence, they are accessed through URLs.[73]

Table 3. *REST vs SOAP Comparison*

REST	SOAP
is an architectural style.	is XML-based message exchange protocol.
uses XML or JSON for communication.	uses Web Services Description Language (WSDL) for communication.
calls services through URLs.	invokes services though Remote Procedure Call (RPC).
transfer is over HTTP.	transfer is over HTTP or SMTP.
requires less bandwidth comparatively.	requires higher bandwidth comparatively.

3. PROPOSED METHODOLGY

This section proposes a methodology that will be used for the system development. In first subsection, a communication model has been proposed based on the RTU, that is selected during this proposal, key criteria for RTU preferences are mentioned in that section. Subsection 2 compares the different webservices and identifies why RESTful webservices are better suited for this implementation. Next section identifies the RESTful APIs that can be used to invoke different services. Since this system is comprised of different subsections and applications, it is important to define inputs and outputs for each system to design a data flow structure for the system. Furthermore, this chapter proposes a criterion that should be used for robot selection for the given application. Last part of the chapter defines a system test methodology and proposes different system integration test approaches.

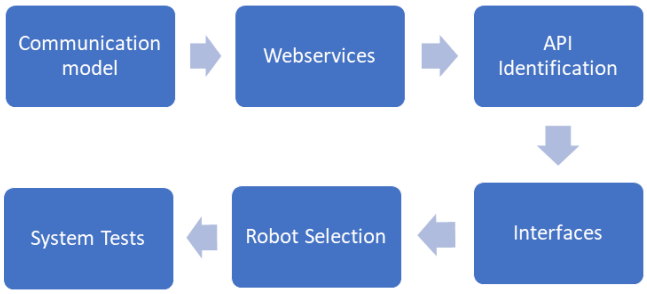


Figure 25. *Proposed Methodology Flow Chart*

3.1 RTU – Robot Communication Model

RTUs or Remote Terminal units are microprocessor-controlled devices that are used in Distributed Control Systems (DCS) to transmit data to master system, and to receive messages from master system to control connected devices. RTUs can execute small programs independently to simplify the process. This thesis proposes to use INICO S1000 RTU device, which is a programmable device and it offers a web-based human-machine interface. Some of the features that make it useful for this implementation are given below:

- S1000 offers logic control using IEC61131-3 Structured Text with built in compiler, which makes it useful to implement simple logics at RTU level.
- It offers web-based HMI with automatic data refreshing capabilities.

- S100 offers built in web server for monitoring and configuration. IT can be used with both REST and SOAP webservices.
- S1000 provides fast Ethernet interface and can be used to communicate over TCP.
- It can be used to monitor events or alarm detection.
- It offers real time control, and web-based programming and configuration.

INICO S1000 can be used for client - server communication model. In client – server communication, service provider is referred as server, while service requester is client. Client is concerned only with the response of its request and does not share any resources with server. While server is the one performing the tasks and delivering response when completed. It is suggested to use client – server model on TCP/IP for Robot - RTU communication.

Communication between client and server only happens when they are in the same internet network. Client – Server communication is bidirectional communication, where client requests the service and server responds to it. Client opens a communication number by addressing the server through IP address and port number. Port number identifies the communication protocol and it defines the communication endpoint.

Once the server has fulfilled client's request and has delivered the service completion message, any of the two devices can close the connection. TCP server can serve multiple clients at a time, but client can communicate with only one server. In case of multiple clients, server allows multiple clients to establish connection. As soon as a client opens a connection, server takes its request and starts a separate sub job for that client.

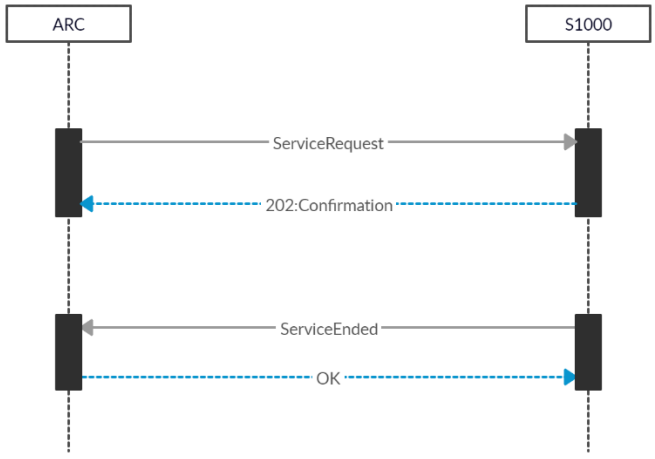


Figure 26. *Sequence Diagram of Orchestrator – S1000 Communication*

Event notification capabilities are important feature of S1000 RTU. It is proposed to request services using the ARC, and when server has provided the requested service, an

event can be published using the RTU, which can be used to notify the operator that now robot is available for next request. An event can also be published when operator makes a request to let the operator know that the request has been received and accepted.

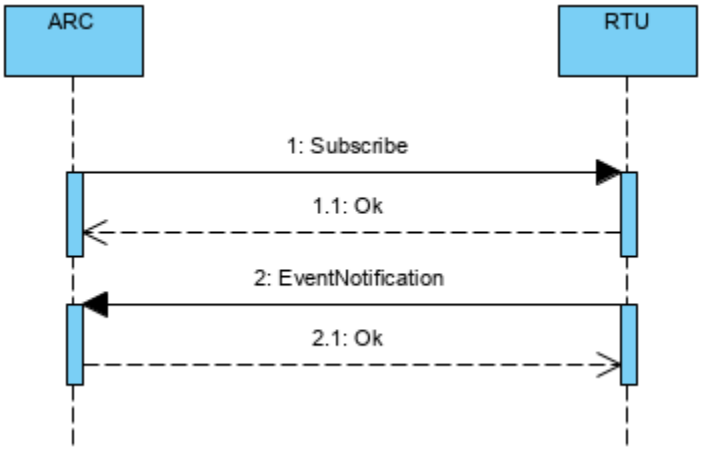


Figure 27. *S1000 - Application Event Notification*

Advanced Rest Client (ARC) is developed by Google and it is an API testing tool. ARC is a REST client that can be used to create and test HTTP requests, it provides full control over the connection and request and response headers. Even though ARC can be used with all HTTP methods, it is only suggested to use GET and POST methods since, rest of the method actions are not useful in this implementation.

GET and POST are HTTP methods, GET is used to request data from a resource, POST is used to send data to update or create a resource. POST request cannot supply entire data through URL, so, it uses binary string to send message or data from client to server, While GET method do not need any additional binary string, entire data is contained in URL.

3.2 Web Services

A detailed description of webservices is given in section 2 of this thesis along with a comparison of REST and SOAP webservices. This subsection suggests using RESTful webservices based on the comparison given earlier, with reasons that make it useful for this implementation.

SOAP is an xml-based messaging protocol for, with a machine processable format interface which is called WSDL. A webservice is defined as XML notation that has all the essential details like message format, transport protocol and location to interact with webservices. On the other hand, REST is architectural style, here the data format is defined using JSON, and it uses the HTTP transport protocols.

REST is a concept, it can use any protocol like SOAP or HTTP, while SOAP being a protocol, cannot use REST. REST can use different data formats such as: HTML, XML, JSON or even plain text. While SOAP can only use XML. REST requires less bandwidth and resources as compared to SOAP. REST is slower and less reliable then SOAP and it can transfer only over HTTP. REST inherits the security from the underlying transport protocol, and it can invoke services through URLs, which makes RESTful webservices more useful in the implementation as compared to SOAP.

3.3 API Identification

API or Application programming Interface are used for communication between different software components or by different applications. Both APIs and webservices are used for communication between client and server. In fact, all the webservices are APIs but it cannot be said the other way around. REST API is a webservice API since, it makes a request over a network using HTTP methods. Components of a REST request are given below:

- URL Path
- HTTP Method: GET and POST are used in this implementation
- Header: The additional information that is sent from the client to server is mentioned in the header, such as content body type.
- Parameter: They define how the resources will be returned.
- Body: Contains the data that needs to be delivered.

To identify the APIs, it is important to identify the services needed. For this implementation, the services needed are listed below

- Pick a pen
- Pick Green Pen
- Pick Red Pen
- Pick Blue Pen
- Change Pen
- Configure Z
- Draw
- Draw 1
- Draw 2
- Till Draw 9
- Place Pen
- Discard Pen

- Publish Events
- Pen Change Started Event
- Pen Change Ended Event
- Draw Started Event
- Draw Ended Event

Pick pen and change pen can be invoked with the same API and if gripper is already holding a pen, it can place it back before going for a new pen. Events are published when server starts providing or ends providing a specific service, these events are listed above. The APIs used for these REST services and HTTP methods used are listed below along with the description of what they can be used for.

Table 4. *REST API List*

Service	API	HTTP Method	Body
Pick Green Pen	http://{RTU_IP}/rest/services/picGreenPen	POST	{"destUrl":""}
Pick Red Pen	http://{RTU_IP}/rest/services/picRedPen	POST	{"destUrl":""}
Pick Blue Pen	http://{RTU_IP}/rest/services/picBluePen	POST	{"destUrl":""}
Draw 1	http://{RTU_IP}/rest/services/Draw1	POST	{"destUrl":""}
Draw 2	http://{RTU_IP}/rest/services/Draw2	POST	{"destUrl":""}
Draw 3	http://{RTU_IP}/rest/services/Draw3	POST	{"destUrl":""}
Draw 4	http://{RTU_IP}/rest/services/Draw4	POST	{"destUrl":""}
Draw 5	http://{RTU_IP}/rest/services/Draw5	POST	{"destUrl":""}
Draw 6	http://{RTU_IP}/rest/services/Draw6	POST	{"destUrl":""}
Draw 7	http://{RTU_IP}/rest/services/Draw7	POST	{"destUrl":""}
Draw 8	http://{RTU_IP}/rest/services/Draw8	POST	{"destUrl":""}
Draw 9	http://{RTU_IP}/rest/services/Draw9	POST	{"destUrl":""}
Configure Z	http://{RTU_IP}/rest/services/ConfigureZ	POST	{"destUrl":""}
Place pen	http://{RTU_IP}/rest/services/PlacePen	POST	{"destUrl":""}
Discard Pen	http://{RTU_IP}/rest/services/DiscardPen	POST	{"destUrl":""}
Pen Change Started	http://{RTU_IP}/rest/events/PenChangeStarted	GET	
Pen Change Ended	http://{RTU_IP}/rest/events/PenChangeEnded	GET	
Draw Started	http://{RTU_IP}/rest/events/DrawStarted	GET	

Draw Ended	http://{RTU_IP}/rest/events/DrawEnded	GET	

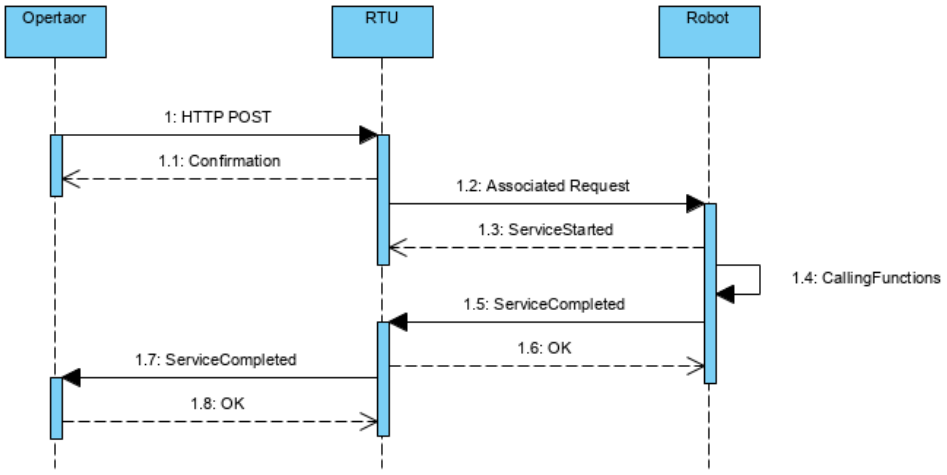


Figure 28. *Communication Sequence Model*

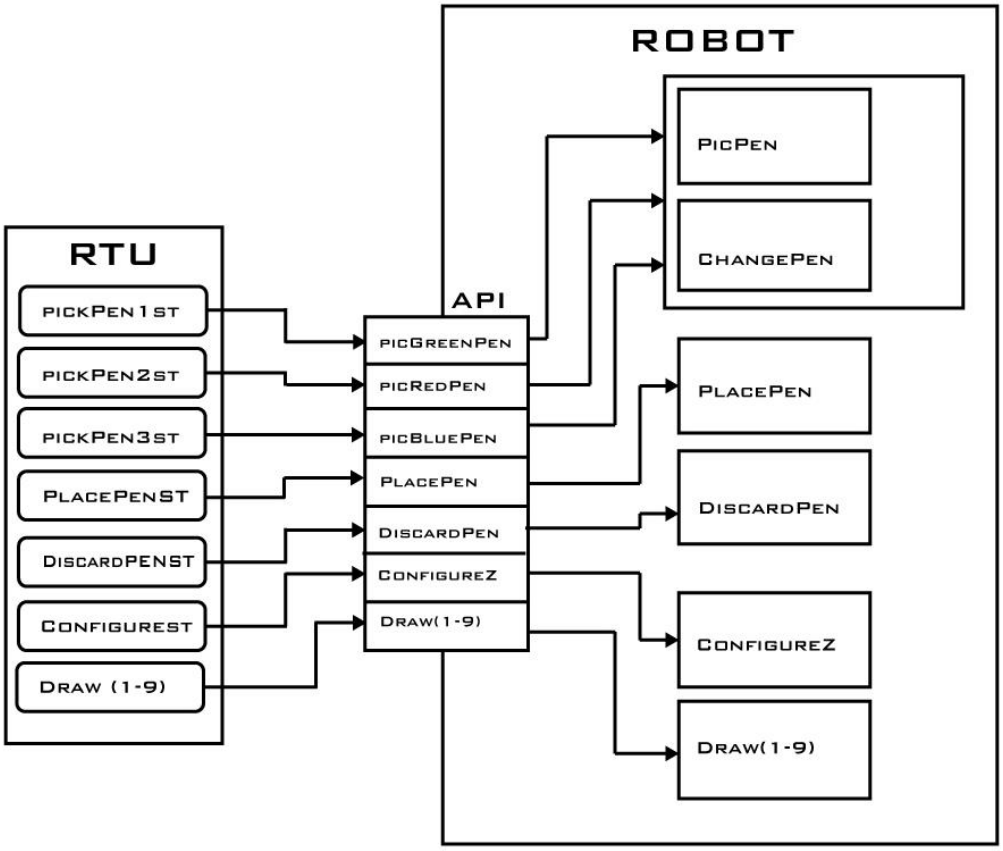


Figure 29. *API Pattern for RTU - Robot Communication*

3.4 Interfaces

There are two main machines used in this process: Robot and RTU, and four different components or programming environments are proposed: robot programming environment, RTU programming environment, environment for free shape algorithm implementation and testing software. The communication between all these applications or software is quite important part of this thesis. This section suggests input and output for each interface.

Robot Programming environment will be used for implementing all the robot related functionalities, from picking a pen to drawing. Robot will be used as a server in this implementation, a TCP server will be created here, that will be listening to a client. Input to the robot will be REST request from client to perform a specific task. The output of the robot will be the actual task required such as drawing. But in addition to that, there are other outputs that will be used as input for other systems or message displays which are given here:

- Inputs to the robot are
  - Instructions from RTU to perform a certain task.
  - Drawing point coordinates from the free shape algorithms and instructions to clear the already existing drawing points on that specific draw point array.
- Outputs of the robot will be
  - Task Starting and task ending messages/notifications or confirmations to RTU.
  - Notifications or messages displayed on the robot teach pendant for the user to monitor the operational activities.
  - An output message over the network socket that is used for receiving the drawing coordinates.

RTU – Robot communication has already discussed in detail in the previous subsection. The RTU is used to send instruction over ethernet, to perform a certain task. RTU inputs and outputs are listed below:

- Inputs to RTU will be
  - Certain request from the user through ARC to perform a certain task.
  - Messages from the robot about starting or finishing a task.
- Outputs of the RTU are proposed as
  - Service requests sent to the robot
  - Event notifications

Free shape algorithm implementation is explained in the section 4 of this thesis. The suggested implementation is different programming environment and drawing coordinates are sent to the robot over a TCP network socket. The output of the algorithm script is the instruction to clear the existing points in the draw array. After that, it will start sending the drawing coordinates over the socket. Input to the script will be a success message after all the points have been received by the robot.

Different tools are used for communication testing purposes, input, outputs and other testing criteria are defined for testing separately. Testing environment and procedure details are discussed in section 5 of this document.

3.5 Robot Topology Selection Criteria

Selection of robot, for automation assembly tasks, is a quite a challenging task. The decision maker needs to select the robot that can perform the given task in lowest possible cost. But there are several other criteria that effect this selection process and over the time, various quantitative methods have been proposed to help identify the best suited robot. The criteria used to select the robots for this task are given below:

- Application

  First step in selecting a robot is to identify the problem or application for which it is needed. Work environment is also an important factor. If the task requires a robot – human collaboration, Cobots are the best option. Gripper or tool selection is also an important part of planning a robot-based application.

- Payload

  Payload is the maximum load, including workpiece and tool/griper, that a robot is going to carry in performing tasks. So, the load capacity of a robot should be higher than payload.

- DOF

  DOF is directly related to number of axis. In performing simple pick and place assembly tasks, a 4-axis robot is enough. But 6 or 7-axi robots are a better option if work involves lots of twisting in small workplace.

- Range

  A robot's workspace is quite an important factor in choosing a robot. Figure 4 in this thesis, shows the work envelop associated with different robot configurations. OMRON ecobra600 robot has a reach of 600mm which is quite enough for this application.

- Positioning Accuracy

  In applications such as welding, drawing or assembling an electronic circuit, quite precise robots are needed that can perform the same task repeatedly with the same positioning accuracy.

- Speed

  Rate, at which a robot should perform its job in an assembly line, also a deciding factor in choosing a robot. Manufacturers usually specify the maximum speed (per second) of each joint of a robot, because during speed varies from zero to maximum. Hence, the acceleration is also important in choosing a robot.

- Body weight

  A robot's weight is an important factor if robot needs to be placed over a rail, conveyor or on another machine.

- Braking System

  Enough brakes on robot's axis ensure accuracy but, it also ensure that robot will lockup in case of accidental power out, and it will not cause any destructions or harm.

- Protection level

  Protection level of a robot is defined by its IP (Ingress Protection Rating) rating. Robot's IP rating requirements depend on the task and environment i.e. different IP ratings are needed for different applications or environment, such as IP67 rating defines that robot can work in dust tight, it can work in dusty environment and, it can be continuously stay underwater as well.

Selected robot will be installed at FASTory assembly line of FAST lab at Tampere University's Hervanta campus. This line is being used by students for academic research and learning purposes. Hence, safety is quite an important aspect of this selection process. SCARA robots are most suited for the assembly line tasks where payload is small. Another reason for selecting SCARA assembly is given as: For best drawing output, pen should be held in parallel to robot's axis, which is quite possible with SCARA. Selection is based on these questions:

- Does this robot have suitable work envelop and enough range?
- Is the robot assembly compatible for this task?
- Even though pen is quite light, is the robot payload reasonable?
- Choosing a robot with way too large payload is not recommended for this application.
- Does this robot offer enough speed and positioning accuracy?
- Does this robot have enough protection level and good braking system to avoid mishaps in emergency situations?

3.6 System Tests

For this implementation, three testing techniques have been suggested.

- Unit testing: unit testing to verify a program or a module as soon as it has been developed. This testing can be manual, since purpose is to validate a block after development.

- Integration testing: Integration testing is more time consuming, and it is done to test the integrated system. Automated integration is recommended since integration testing is more time and money consuming.
- Post development Beta testing: Which will be manual in this case since, it is demo of this thesis.

Before defining test cases, it is important to define the system modules, how they will be integrated and the process flow. Then the modules which will be tested during unit testing will be selected. Inputs and outputs of each system have already been defined in the previous sections. This section first defines the system modules and information flow, then it describes the tests that will be performed as part of system development.

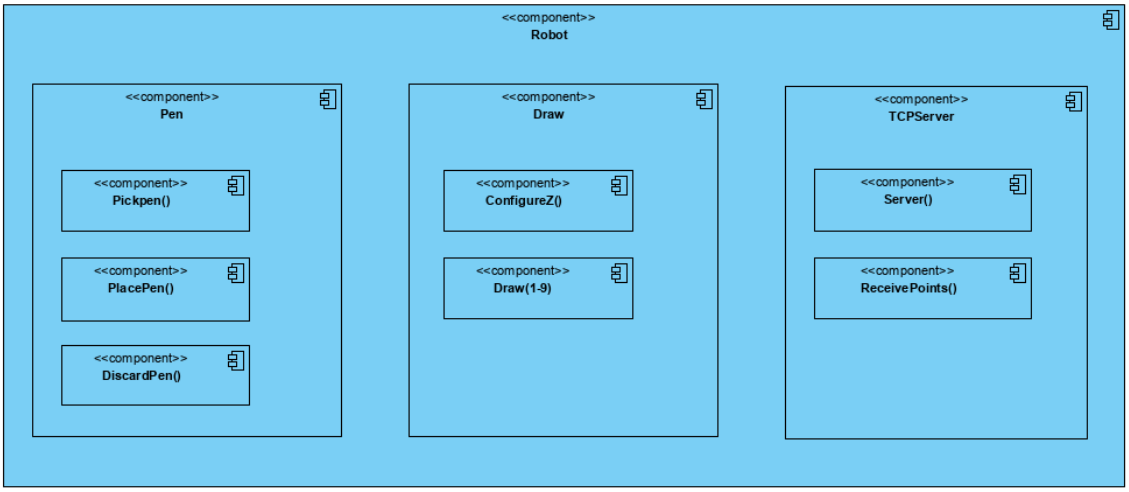


Figure 30. *Robot Modules*

The main modules of the robot are shown in figure 29, server function in TCP server will be able to communicate with s1000 as shown in figure 28, meanwhile it will be able to receive draw point coordinates over the network socket.

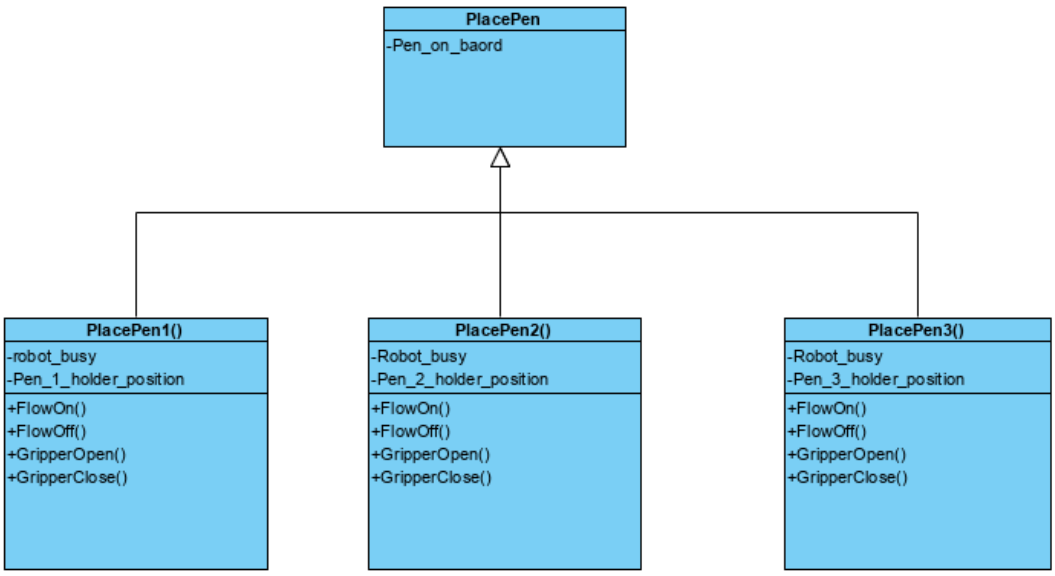


Figure 31. *Pen Placement Class Diagram*

PlacePen class is super class, that is used to decide which pen is being hold at the time, and it will make the decision to class the either of three classes to place then pen based on that. All the functions or classes are called through server(), based on the received request. It is shown in the next figure.

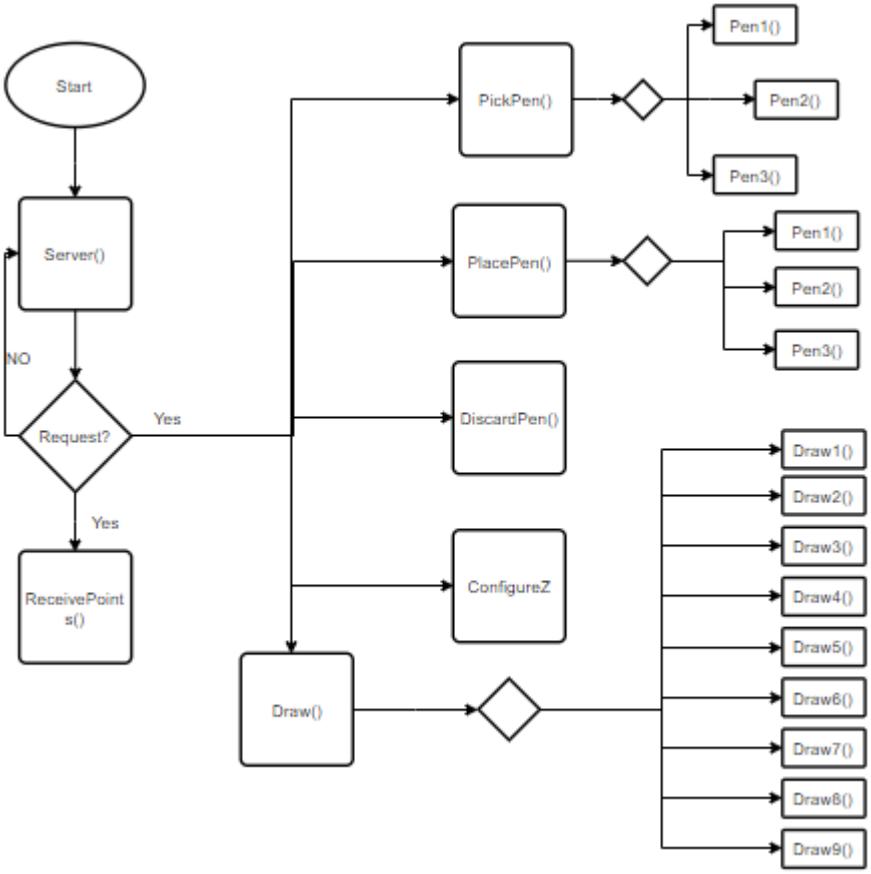


Figure 32. *Process Flow*

Individual functions used for picking and placing a certain number pen can be tested via unit tests, but they will also be tested after the system has been integrated. State based unit testing will be used to test that system is operating in the required manner i.e. it is giving desired output to defined input. Integration testing will be interaction based to make sure that different system components are communicating in the required manner and system's decision making will be verified in integration testing. System Integration tests has been described in detail in section 5. Following testing strategy will be used for system integration testing:

- Perform integration tests on the system
- If something fails, use Unit tests to figure out the cause
- Fix the errors
- Repeat the process

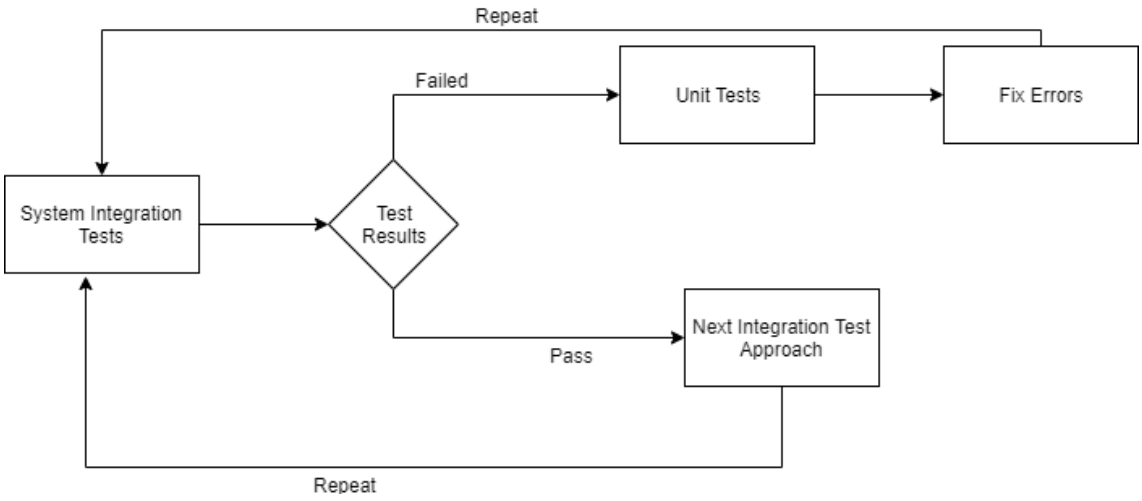


Figure 33. *Proposed Test Approach*

Bottom-up breadth first and Top Down depth first integration techniques will be used for testing. Bottom-up BF testing starts at lower level and test driver will used to call certain operation in case needed. This testing technique will be used to test all the modules to verify the different system conditions that have been defined. While in Top Down, testing starts at top, this technique will be used to test system flow as it will be used in the actual operation. Stubs will be used to provide required resources to the top modules. Following figures show the test flow for both techniques.

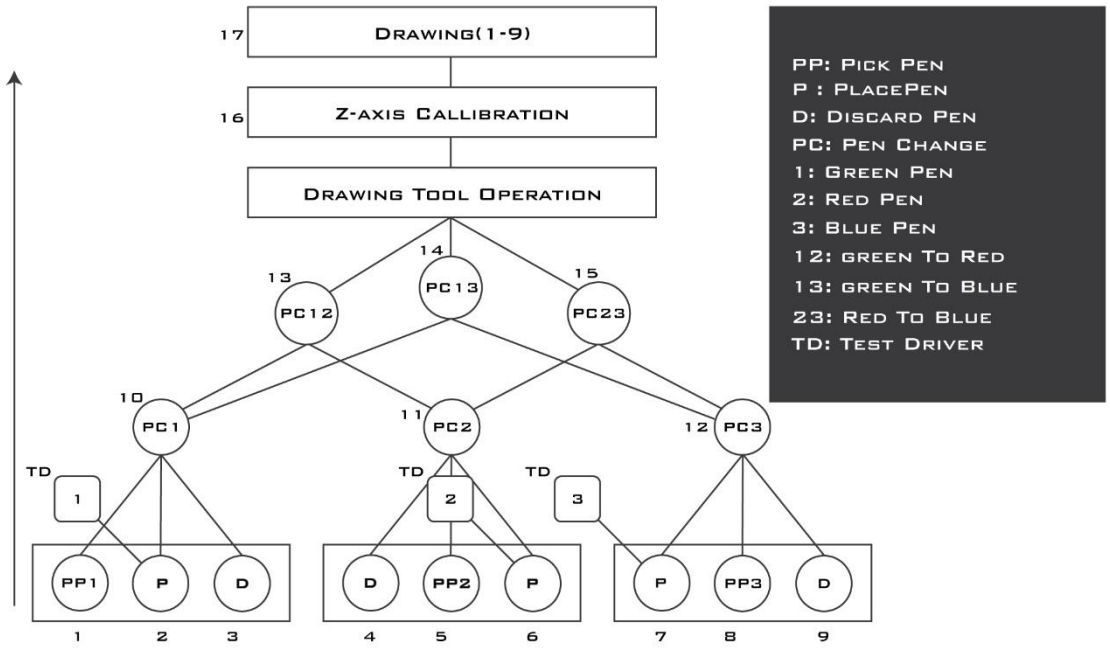


Figure 34. *Bottom-up BF*

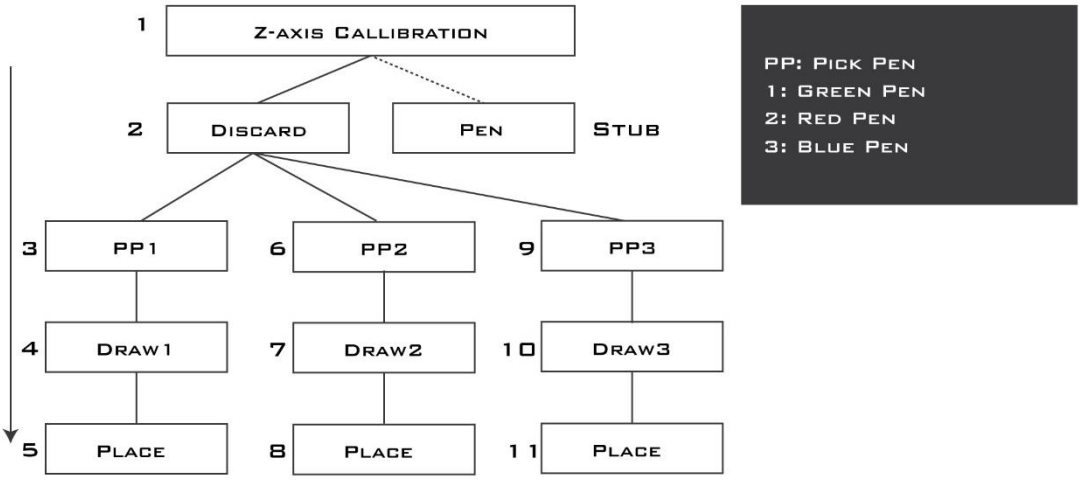


Figure 35. *Top Down DF*

4. IMPLEMENTATION

4.1 FASTory Assembly Line Description

FASTory is an assembly line installed at FAST lab of Tampere university's Hervanta campus. This line has 12 workstations, 10 of which are identical; these identical workstations are used to draw mobile models. WS7 is used to load and unload pallets, While the last workstation, which is WS1, loads and unloads papers on the pallets. Each WS on this line offers 9 different type of designs in 3 different possible colours. FASTory robots draw frames, screens and keypads of mobiles, offering 9 variance of each part which makes FASTory able to produce 729 different products. Each WS has 2 types of conveyors. Main conveyor which is used if WS is idle or pallet requires design from the that cell. Bypass conveyor is used in case the WS is busy or no services are required from that station.

Figure 36. *FASTory Work Cells at FAST Lab – TUNI*

FASTory has been a testbed for several EU projects such as eSONIA, eScop, ASTUTE and C2NET. [79]

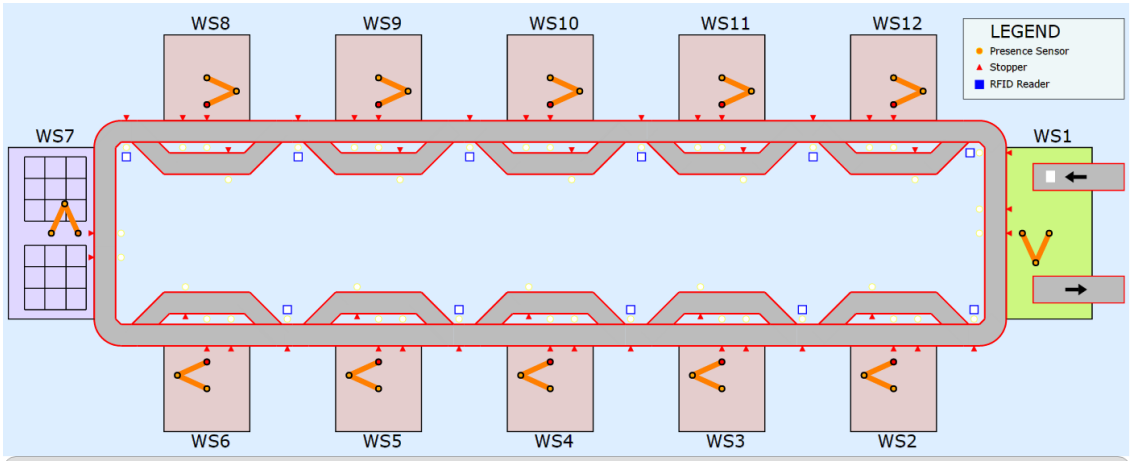


Figure 37. *FASTory Layout*

All identical WS have robots, which is the main element of each cell. Most of these robots are caged because they are industrial robots. All the robots installed at FASTory were Sony SRX-611 SCARA robots. These robots have 4 joints, 3 revolute joints and 1 prismatic joint, and 4 DOF which is represented by x, y, z coordinates and rotation around z- axis i.e. (x, y, z, R_z). These Robots have custom made grippers which can hold a pen, structure and working of these grippers is explained in the section 4 of this thesis. Cells also have custom made pen holders in each cell, that has the capacity to store 3 pens at a time.

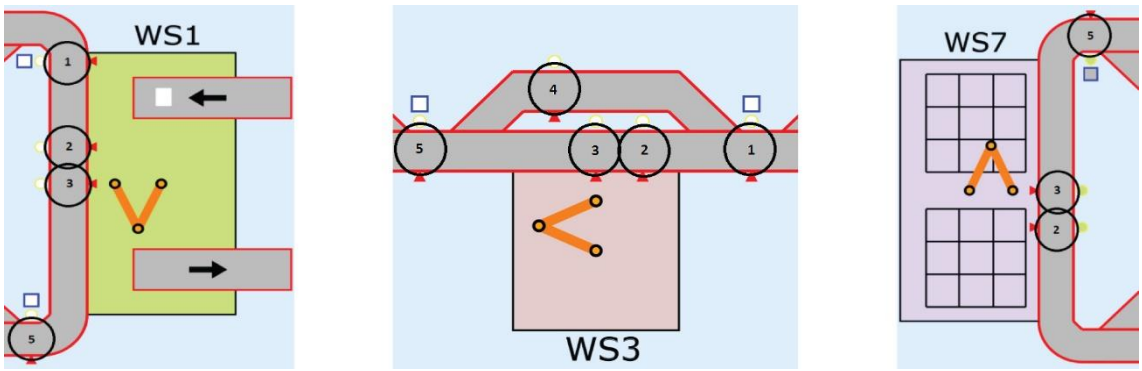


Figure 38. *Layout of WS 1, 3 & 7*

Rest of the WS are identical to workstation 3. Numbers on the conveyor represent the five different zones, all these zones have sensors to detect the presence of pallet. All the pallets have unique RFID tags, which are read by RFID readers located on each workstation. RFID are installed at zone 1 of each station and they are used to identify the pallet and stoppers at each zone are used to stop the pallet in that zone if needed, while the rest of the pallets keep moving. Figure 38 shows that WS1 does not have zone 4, while WS7 lacks zone 1 and 4. The functionality of these zones is described the table 5.

Table 5. *FASTory Conveyor's Zone Description*

Zone 1	This zone is used for decision making, whether the pallet will be transferred to this WS or next. RFID readers installed at this zone helps to identify the pallet.
Zone 2	This zone is used for queuing the extra pallet, before it is transferred to next zone. When the zone 3 is free, pallet is moves from zone 2 to zone 3.
Zone 3	Zone 3 is used to perform the actual task. Pallet stays in this zone till the robot has finished drawing.
Zone 4	Zone 4 is used to bypass the WS. This zone makes decision from which zone (3 or 4) pallet will move to zone 5, if zone 5 is empty.
Zone 5	This is the exit zone for pallet to leave that WS.

Actuators installed at FASTory are pneumatic actuators, which are controlled by an electronic valve. Now a days, a few changes has been made in the line, WS7 has been removed and Sony SCARA robots installed at workstation 3, 5 and 8 have been removed with SCARA robots SDA20D from KUKA and OMRON manufacturers. Now, WS1 is equipped with dual arm Yaskawa Motoman robot designed for complex assembly tasks. This robot has 15-axis; and this is the first robot that has introduced FASTory line to collaborative robotics.

Robots and conveyors installed at this assembly line are controlled through Remote Terminal Units (RTUs), these RTUs communicate via local network, which is Ethernet in this case. RTU used in this thesis is INICO S1000, which is a programmable device. S1000 offers web-based Human-Machine interface (HMI), It can be programmed using DPWS and RESTful webservices to control the process. Each of these devices, which connected through Ethernet, have their unique IP, but same subnet mask which is 255.255.0.0.

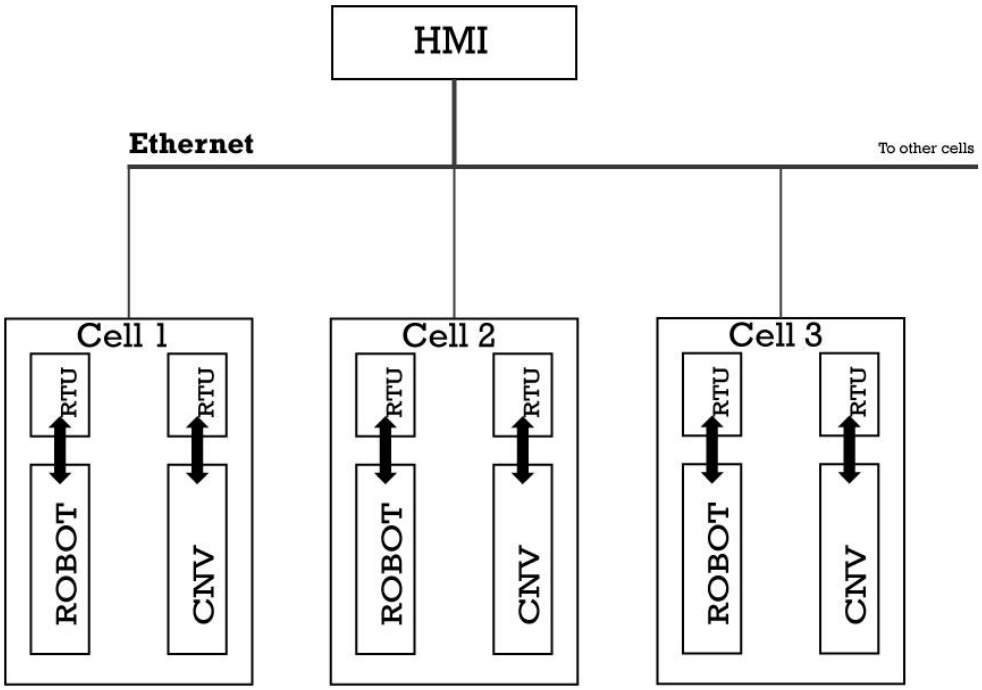


Figure 39. *Assembly Line's Network Configuration*

Based on the comparison given in table 3 of section 2, webservices used in this thesis are RESTful. REST uses XML or JSON. Since, all requests in this communication are independent of each other, and no data is requested from previous requests, suggests that REST is a better choice since, it is stateless.

The communication between HMI and S1000 is only using GET and POST methods. HMI used in this case is Advanced Rest Client (ARC), which is developed by Google. ARC is a REST client that can be used to create and test HTTP requests. When a request is made though ARC, RTU responds to the application with an empty message which is 202 or 404. 202 is used for accepted requests and 404 for forbidden. RTU forwards this request to robot, and when request is completed RTU sends a confirmation message through POST method, which is an empty message. Messages are sent on top of TCP/IP using client-server protocol.

4.2 Cell Description

Robots used in this thesis are ecbora600 pro and KR3 R540. Ecobro600 is a SCARA robot, while KR3 is 6-axis a robot. These robot topologies are fit for this implementation since, they offer enough accuracy, payload is quite light in this application and work envelope satisfies the requirement of this job.

Table 6. *Comparison Between ecobra600 and kr3*

	OMRON ecobra600 pro		KR3 R540	
Number of Axis	4		6	
Number of Controlled axes	4		6	
Rated Payload	5.5 kg		2 kg	
Maximum Reach	600 mm		541 mm	
Work Envelope	Figure-40		Figure-41	
Controller	eAIB		KR C4 Compact	
Accuracy/Repeatability	xy	±0.017 mm	±0.02 mm	
	z	0.003 mm		
	theta	±0.019°		
Joint speed	J1	386°/s	A1	530°/s
	J2	720°/s	A2	529°/s
	J3	1100 mm/s	A3	538°/s
	J4	1200°/s	A4	600°/s
	-	-	A5	600°/s
	-	-	A6	800°/s
Joint Range	J1	±105°	A1	±170°
	J2	±157.5°	A2	-170°/50°
	J3	210 mm	A3	-110°/155°
	J4	±360°	A4	±175°
	-	-	A5	±120°
	-	-	A6	±350°
Programming Environment	ACE		WorkVisual	
Programming Language	eV+		KRL	
Weight	41 kg		26.5 kg	
Protection Rating	IP20		IP40	
Mounting Position	Floor		Floor, Ceiling, Wall	

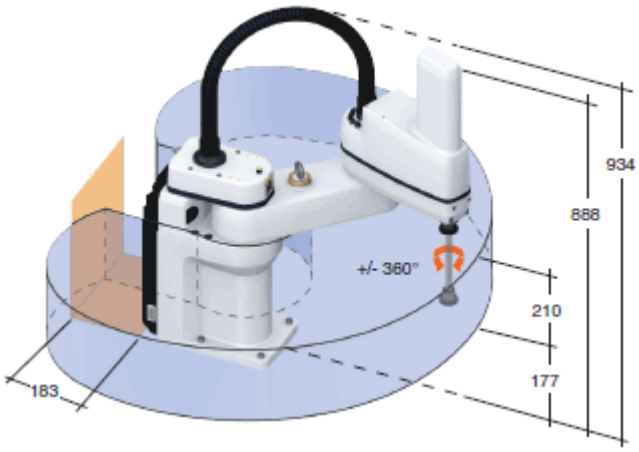


Figure 40. *ecobra600 Work Envelope [80]*

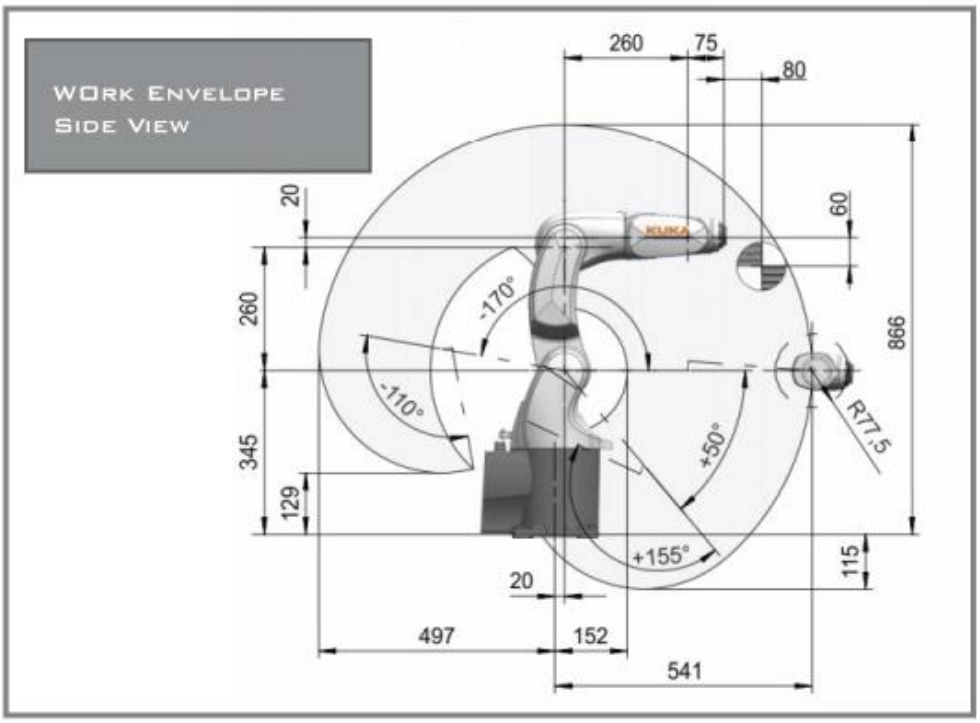


Figure 41. *KR3 R540 Work Envelope, Side View [81]*

Based on the comparison, it can be concluded that both robots can be used for the given application. They are suitable to work in the target environment; work envelope covers the desired locations in the work cell. They have enough joint speed and accuracy to follow the given path efficiently and accurately.

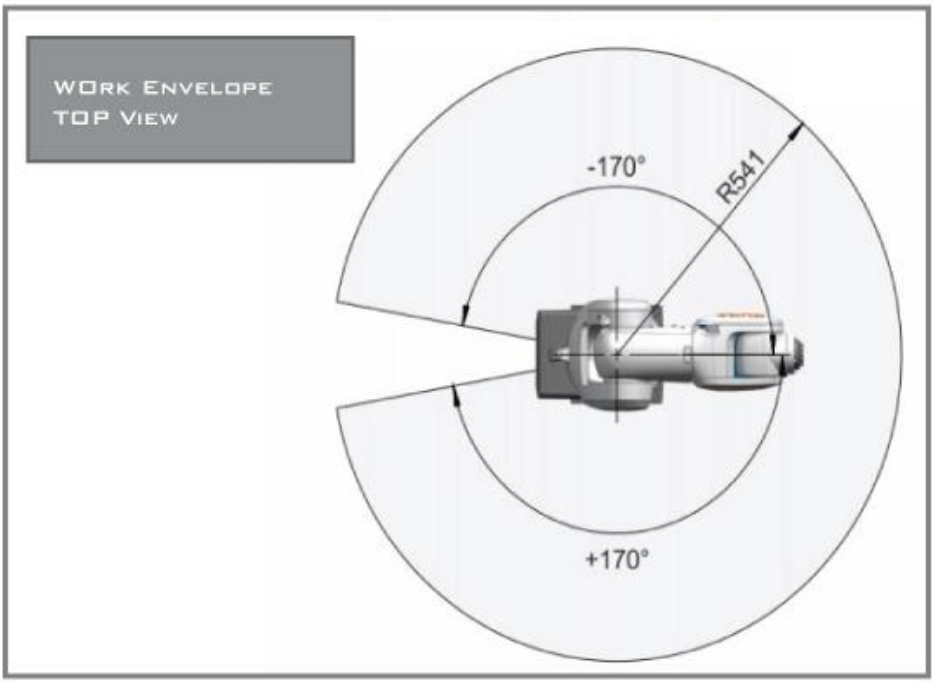


Figure 42. *KR3 R540 Work Envelop, Top View [84]*

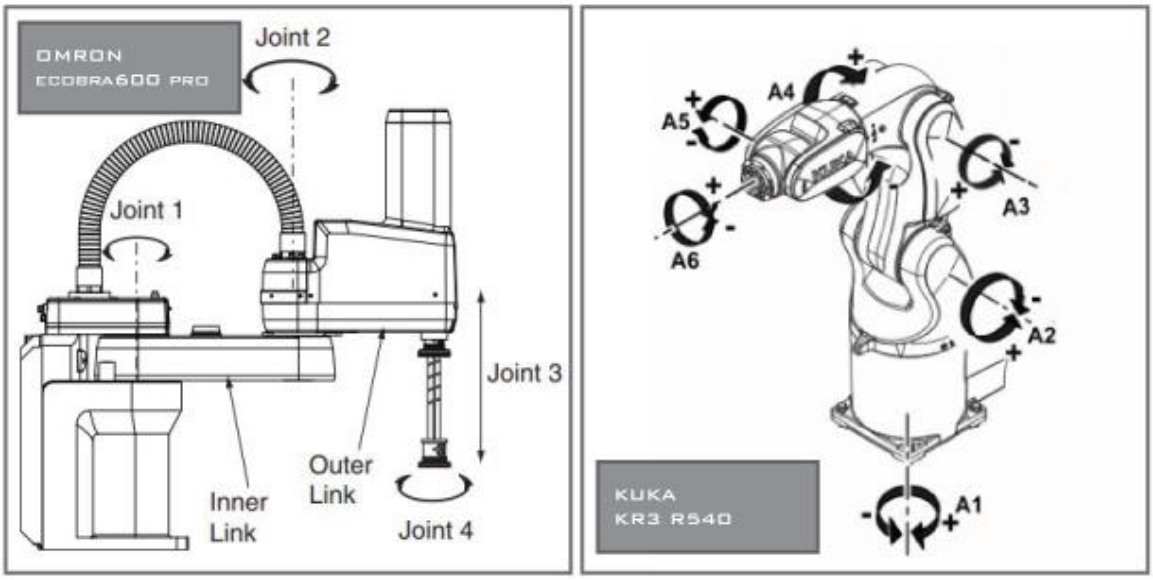


Figure 43. *Robot Axes Rotation Direction*

4.2.1 **OMRON ecobra600 Pro**

As stated before, OMRON ecobra600 is industrial SCARA robot with 4 joints, installed at cell number 8 of FASTory assembly line. Figure 43 shows that joints 1,2 and 4 are rotational, while joint 3 is translational, joint ranges are given in Table-7. The robot is attached to built-in eAIB controller, which is mounted on the base of the robot. Hence, it can be used as a standalone robot without any need for external controllers, servo controls and amplifiers are contained in eAIB. Cell 8 is also equipped with a T20 pendant

that provides manual control of joints for jogging to different locations. A Front panel is also attached to robot that has three controls: a key for switching between manual and computerized mode, a button for high power control and E-stop safety button, as shown in Figure 45.

Figure 44. *ecobra600 pro with eAIB Controller*

If more features and connectivity are needed, eCobra robots can be integrated a SmartController EX motion controller. More vision support can be added by adding SmartVision support.

eAIB is programmed using Automation Control Environment (ACE) using eV+ programming language. ACE have total control of the controller, it can be used for jogging, IO monitoring, programming, debugging and executing purposes. It can be used for online or offline programming, it has 3 main windows: Workspace Explorer, Editor window, and 3D virtual display. Workplace explorer can be used to explore or write V+ programming files or writing, to save location, variables can be created or updated here; ACE workplace explorer interface can also be used for C# programming.

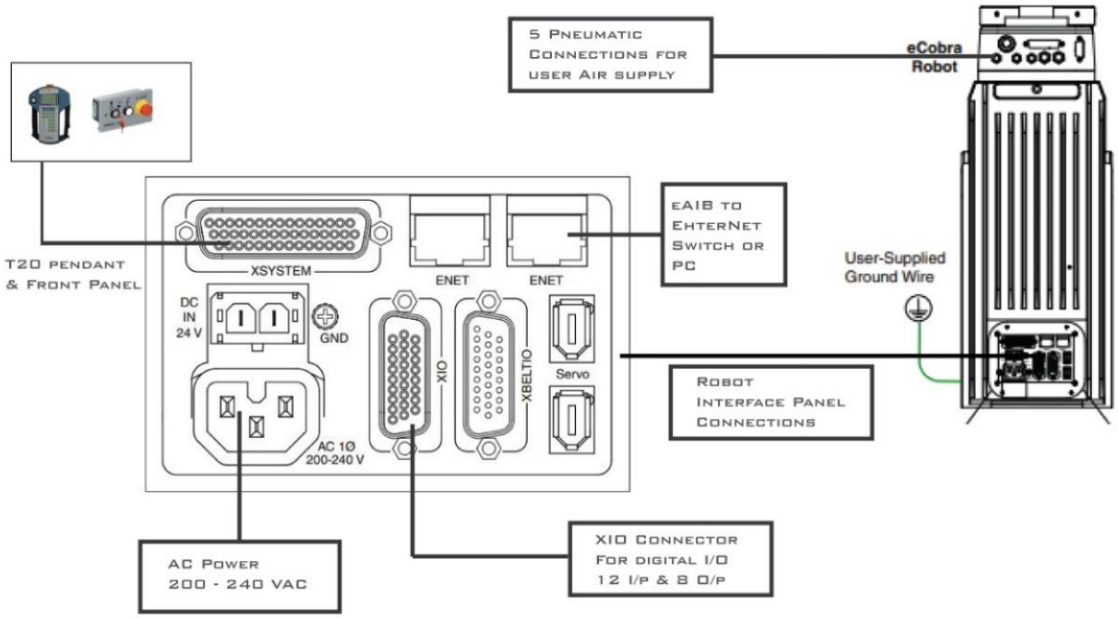


Figure 45. *Robot Interface Panel*

Robot can be connected to NJ/NX Machine automation controller for IEC61131-3 programming; this controller can be used to control operations and program the robot directly from a PLC. Controller is connected to s-1000 or any other device at FAST lab through a HP switch. The pneumatic connections at the back of the gripper are internally connected to the air connectors at Joint-2, from where they are attached to the gripper.

Figure 46. *HP Ethernet Switch*

4.2.2 **KUKA KR3 R540**

KR3 is a 6 joint anthropomorphic manipulator, installed at cell number 3. The robot is attached to an external controller KR C4 Compact.

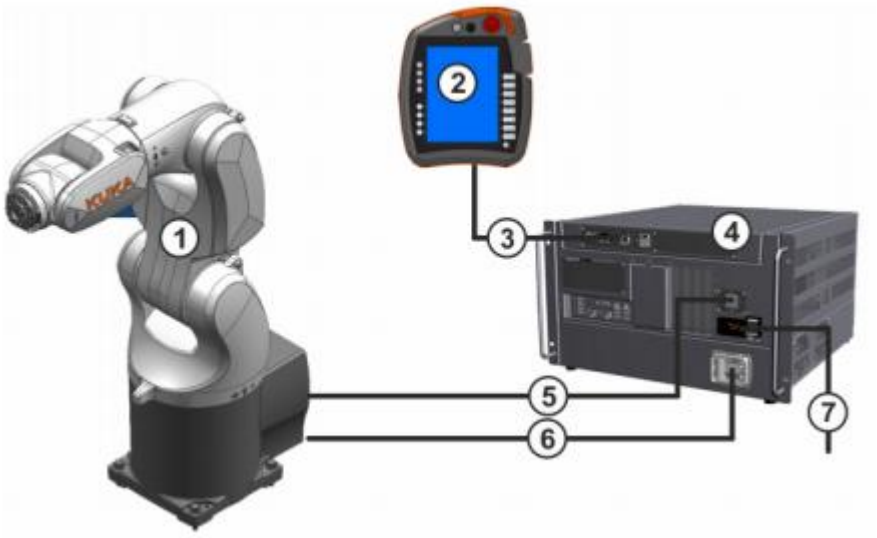


Figure 47. *System Cable Diagram*

- 1 is Manipulator which is attached to KR C4 controller (shown by number 4) through cable 5 and 6 which are data and motor cables respectively.
- 3 is a connection cable that connects SmartPAD control panel (number 2) to control panel.
- Controller can be attached to pc or s-1000 through cable 7, which is an ethernet connection cable port.

SmartPAD can used for jogging the robot, but it has a lot more functions as compared to ecobra's T20 Pendant. Smart controller can be used for inline programming and to execute a program. It has all the required functions for programming and operating an industrial robot. It has KUKA System Software (KSS) installed for operations and control.

WorkVisual can be used for programming and debugging purposes. It is a software interface from KUKA, it can be used for I/O mapping, and it can import or update the files on SmartPAD. KUKA Robot Language (KRL) can be used to program KUKA robots. EthernetnetKRL can be installed on the controller for TCP/IP communication to use it as a server for s-1000, which is main objective of this thesis.

4.2.3 **Gripper**

FASTory robots are equipped with custom made grippers to hold the pens. These Grippers have 1 mechanical slider for vertical motion and two fingers to hold the pen. End-effector is pneumatic powered and air supply is controlled via external SMC solenoid valves. Solenoid are powered via output ports of the robot.

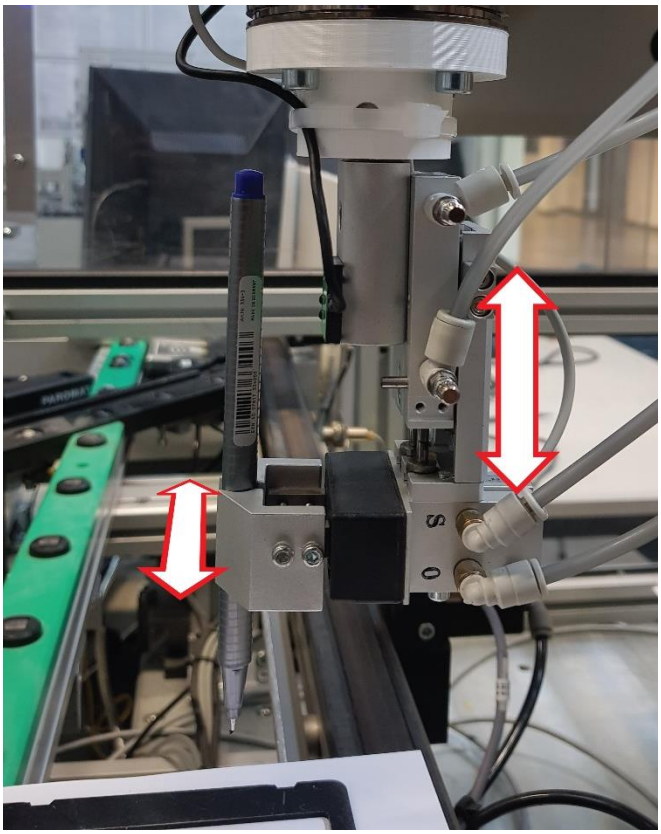


Figure 48. *End-Effector Movement Directions*

Gripper is equipped with 3 electrical sensors, which also have a red light for visual feedback.

- UM-R5TVP is a photoelectric sensor, it is used to verify the presence of pen inside the gripper finger.
- SMC – DA93 is a Reed switch, it works as a position sensor for SMC – MXS6 – 30 slide table. It is used to calibrate the z-axis of gripper for drawing.
- D - M9P is a solid-state auto switch, it is used as a feedback for gripper fingers to verify that gripper is open or close.

Pen holders are placed inside the cell, within the reach of the robots, so, they can pick and place the pens on instruction to start drawing. S1000 and solenoid valves have been added in the control box of the robot, which is below the floor on which robot is mounted. As stated in the section 3, S1000 is an RTU, which supports RESTful webservices and can be integrated with Service-Oriented-Architecture. It is being used as a TCP client with the robot server.

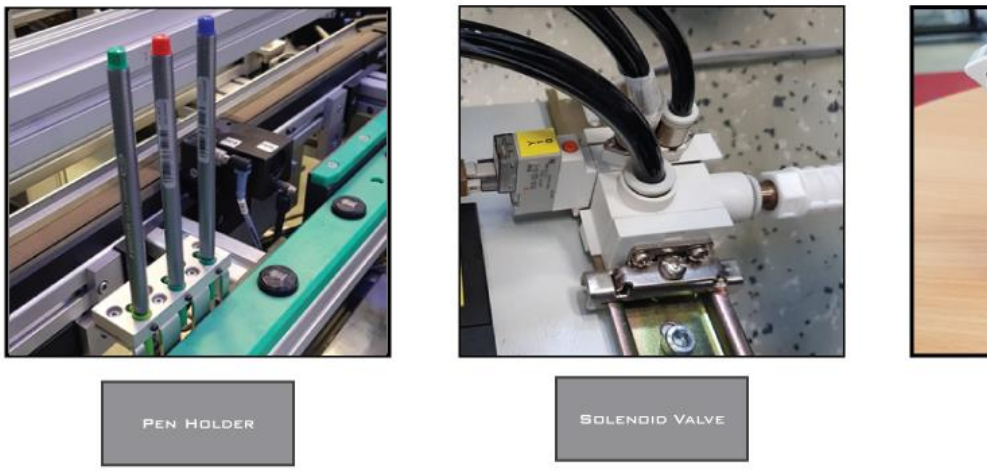


Figure 49. *Cell Components*

4.3 RTU Communication

FASTory can be programmed in using REST or SOAP webservices, but as described earlier, the implementation used in this thesis is based on RESTful webservices. The robots and RTUs are communicating over internet using and they are addressed through their IP addresses.

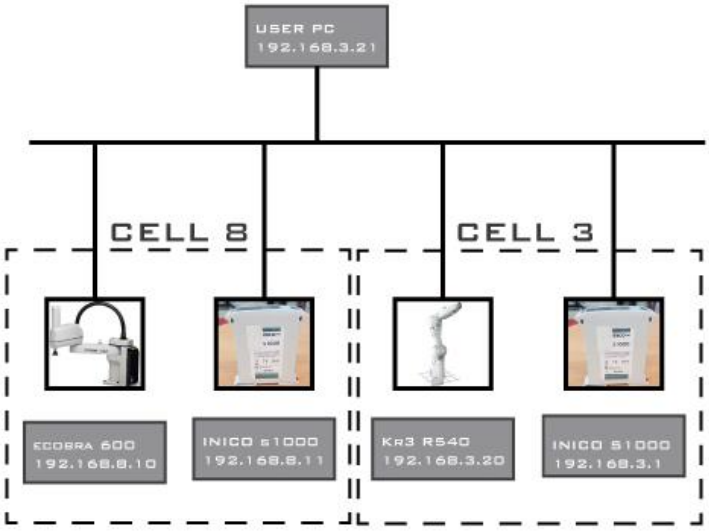


Figure 50. *Network Configuration*

S1000 interface can be accessed using it's IP address. S1000 has 2 modes, configuration mode and run mode. All the changes are made in configuration mode. There are 4 steps that are followed to configure the device for this thesis:

1. Network tab is used to configure the IP address of the s1000 device.

2. I/O tab is used to configure the I/O modules. S1000 device usually comes with pre-configured I/O modules. Net Connection module of I/O tab was used to configure it as a TCP client. The module is configured by assigning an alias which is netconn for this implementation, server IP address and server's port number. One digital output from robot is fed into s1000 as input to track the status of the robot. The I/O is named as "DI_robot_busy".

3. Logic tab is used for logic programming, which is written in ST. During this implementation, one ST program function is used for one request only, So, in total, there are 16 functions: 9 programs for drawing figure selection, 3 programs for pen selections, 2 for placing or discarding the pen, 1 for configuring z-axis and last program is called ROBIO.

    a. ROBIO isn't assigned an alias, because it is being used to receive a message from the robot. The received message is then used to publish events that describe if robot is free or busy. User can monitor/subscribe to these events to verify the robot activities.

       Events are also assigned aliases in REST tab. User can use the following request to access all the events

       ```
       GET: S1000_IP/rest/events
       ```

    b. Rest of the 13 programs are used to request robot to perform certain activity. All these programs are assigned an alias. A user can request a function by using post method

       ```
       POST: {S1000_IP}/rest/services/{alias_name}
       Body: {"destUrl":"192.168.3.21:8080"}
       ```

       Aliases are names after the function. Whenever user post a request, ST program verifies the robot's status

       ```
       IF (DI_robot_busy = FALSE) THEN
            netconn_open( netconn );
       ```

       If robot is free, then it s1000 opens the TCP connection with server and unique string that indicates the services which is requested.

4. Last step in s1000 configuration is to configure the REST services. In REST tab, services are created with certain names which are called alias. Each alias is linked to ST program, that contains a logic. Whenever an alias is called through POST method, linked ST program logic is executed, which sends a message to robot.

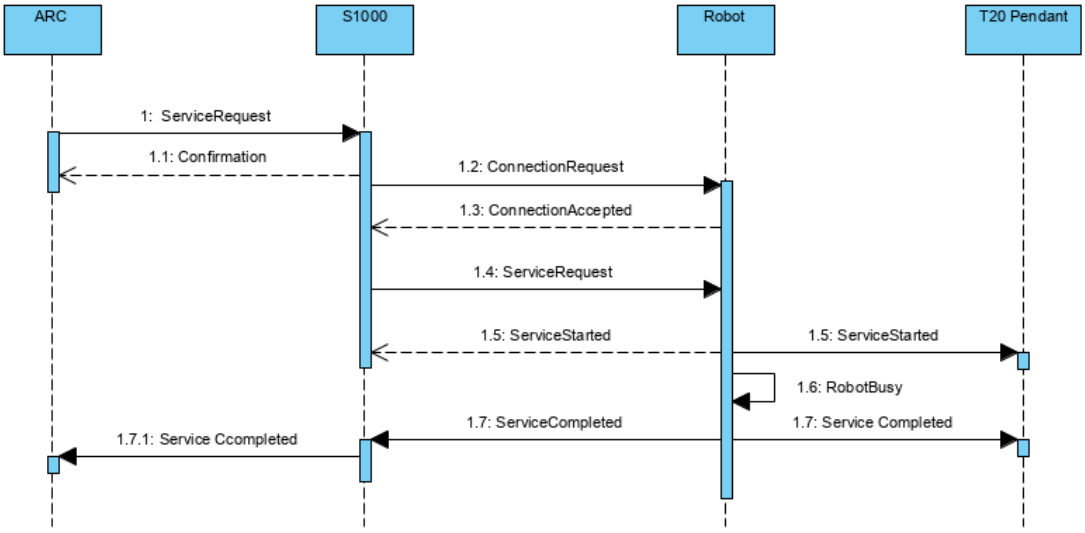


Figure 51. *Communication Sequence*

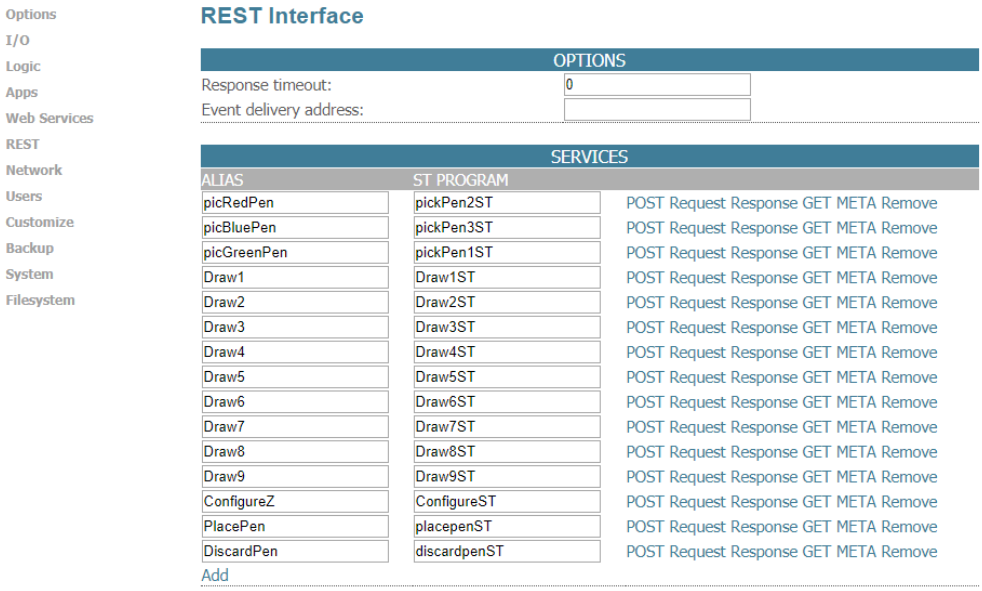


Figure 52. *RTU's REST Interface*

4.4 Robot Functionalities

As stated before, the robots have custom made grippers, designed specially to hold the pens. Since, the paper is placed in XY plane so, it is important that pen is held in z plane. The pen is held in a way that it is parallel to the axis of rotation of last joint and parallel to the axis of robot itself.

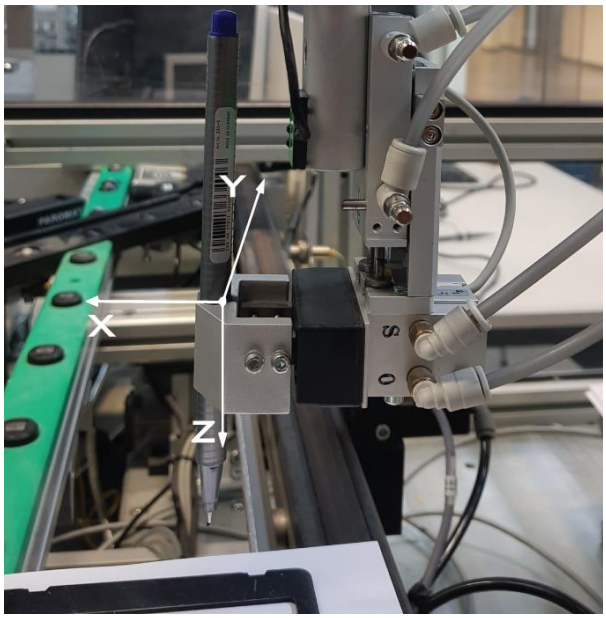


Figure 53. *End-Effector Orientation*

Ecobra600 was programmed using eV+ language in ARC environment. The robot had 2 kind of files V2 files that contained and VAR files that contained data such as constants, variables and locations. On the other hand, KUKA was programmed in WorkVisual environment using KRL language. KUKA's files were also divided in categories: SRC files that contained the actual programs, and DAT files that are called data lists and they contain data. SRC file along with its DAT file are called a module.

Ecobra's code is categorised in 3 main functions: server function that is used for creating and receiving data over TCP server, main function that contains all the task programs and third function called auto. Auto is a default name for OMRON programs, which is used to state the programs that will start running automatically when robot is started. This way, there is no need to start the task manually. While for KUKA, each program has different SRC file and user must start the TCP server from SmartPAD every time controller is restarted. A component diagram is shown in Figure 54, for ecobra600.

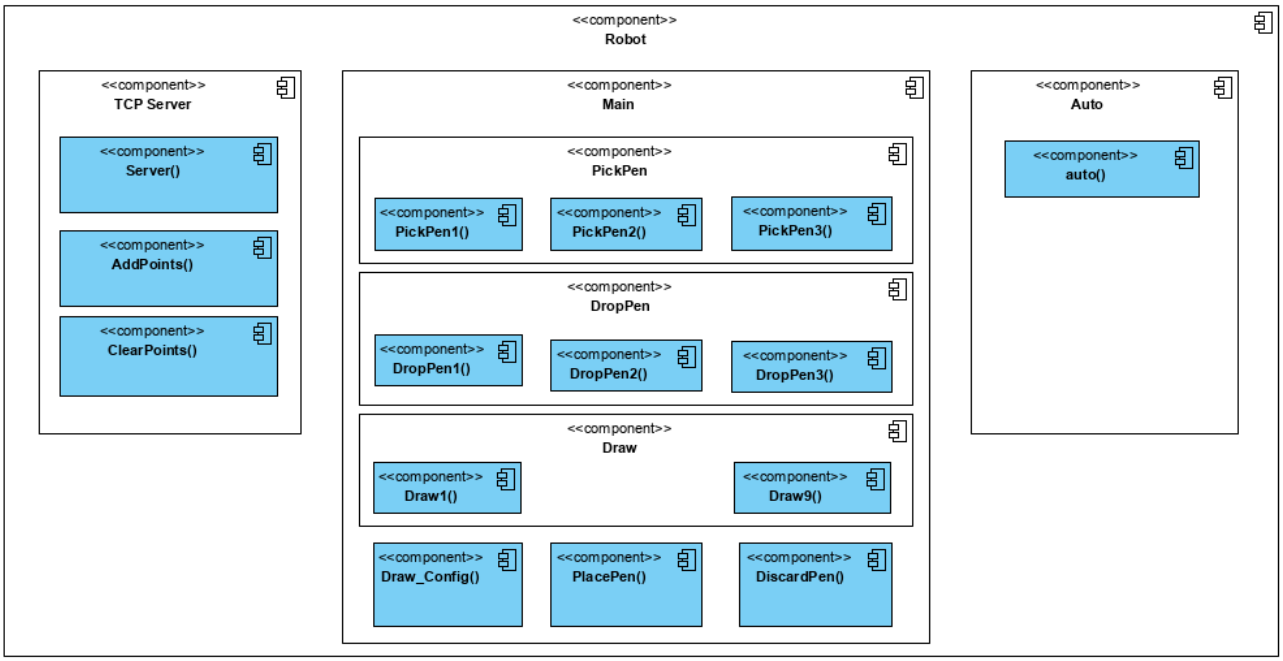


Figure 54. *Main Robot Program Modules*

4.4.1 **TCP Server**

TCP server has been created receive requests from client. When Server is started, it keeps on waiting for a request. When a request is made, it can call one of these functions: PickPen1, PickPen2, PickPen3, Draw1, Draw2 till Draw9 and then DrawConfigure function. Server is also used to receive draw points from free shape algorithm implemented in python, it is explained in next section.

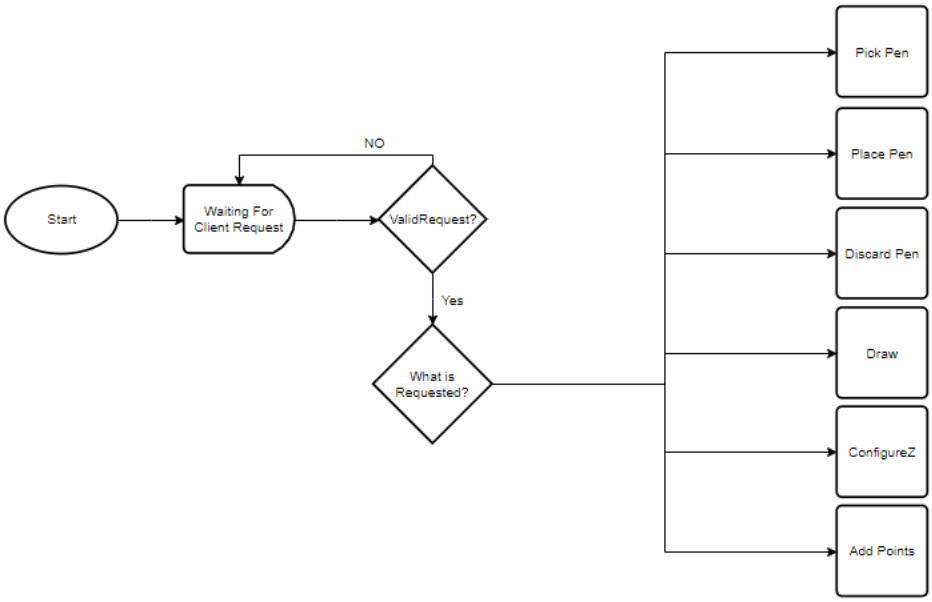


Figure 55. *Figure 1 Decision Making by TCP Server*

Whenever a request is received by the server, if it is valid request it is used calls to call one of those functions. There are 3 decisions for picking a pen, 9 decision for drawing and 9 for adding draw points.

When a function is called through server, robot executes it and when the execution is completed, it writes back to RTU that request has been completed and it notifies the user through T20 pendant. ‘rob.run’ states if robot is busy or free, if robot is busy then rob.run is true.

```
rob.run = TRUE
WAIT (rob.run == FALSE)
WRITE (lun, handle) $ENCODE("pickPenEnded")
PDNT.NOTIFY "Info", "Pen 3 picked"
```

When robot is started, it checks if there is already a pen on board, if not, it marks global variable Pen_on_board as zero. Then it creates a server and starts waiting for the client request. Ecobra robot has a normally open rely on XDIO connector, high power must be provided to close that. Robot has a braking system that stops the robot in case of emergency. Joint 3 brakes are electromechanical, and they are controlled by high power button. Whenever high-power button is on, brake is released. In addition to that, robot has an E-stop button for emergency cases, button is mounted on the front panel, along with high power button and a switch for manual or automatic mode. Whenever E-stop button is pressed, robot’s high power is disabled, operator can enable it manually after releasing the E-stop button through front panel, or it is also possible to do get high power from virtual front panel provided in the ACE software.

After initializing the server, robot tells the operator to enable the high-power, and it waits till operator has enabled it. Now, robot is ready to receive requests from client. After receiving a request from the client, it acts based on the client’s request.

In case, client has requested for it to pick a pen by sending a string “picGreenPen#”, it checks if high-power is enabled or not. If high-power is enabled, it moves forward. If not, it sends back a message to RTU that says: “error”. After sending a message, it notifies the operator through T20 pendant that about the problem. In case, high-power was enables, server executes the relevant function on task 1.

eAIB controller can execute multiple tasks in parallel, for this implementation, server is running on task 2, error handler is running on task 3. Error handler notifies the operator about E-stop error and specifies if the E-stop button should be released from the T20

pendant or front panel. When E-stop is released, it reminds the operator to turn enable the high power by printing a message on T20 pendant.

Apart from executing the picking pen function, server responds back to RTU with a message “pickPenStarted”, and it also notifies the operator via pendant. After calling the pick pen function on task 1, server, which is running on task 2, waits till robot is busy. As soon as, robot finishes picking up a pen, it informs the RTU by sending a string and it also notifies the operator. RTU receives that message and publishes an event to which user can subscribe. RTU also receives a message and publishes the event when robot starts picking up a pen, process is shown in figure 56.

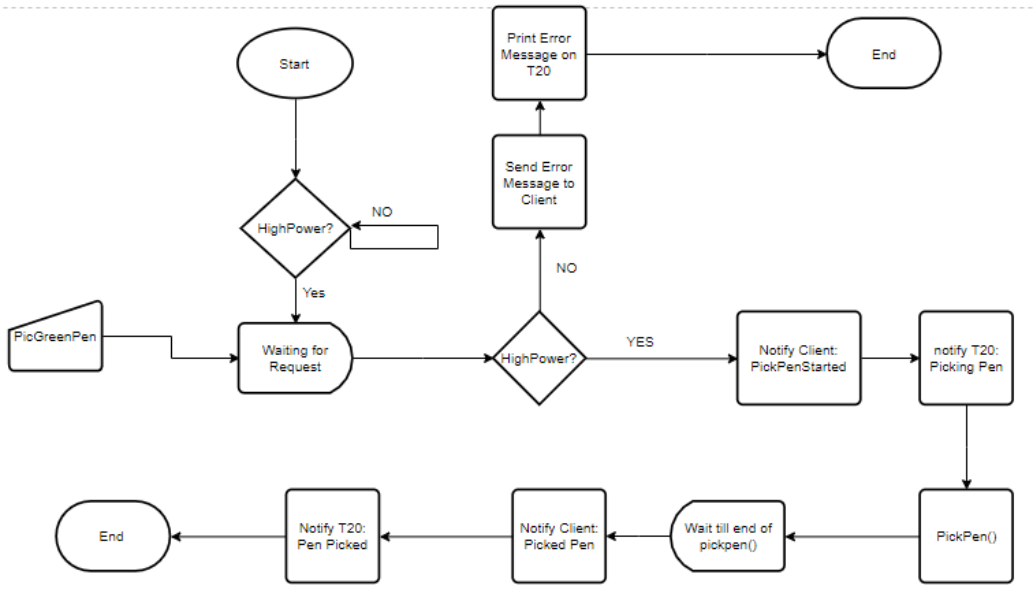


Figure 56. *Flow Chart for Pen Pickup Request*

In the figure 56, only one pen pickup function is shown, in actual, there are 3 programs for picking up a pen. Each colour pen has a different program, figure 57 shows the activity diagram with the actual names of the functions the have been used for each colour pen.

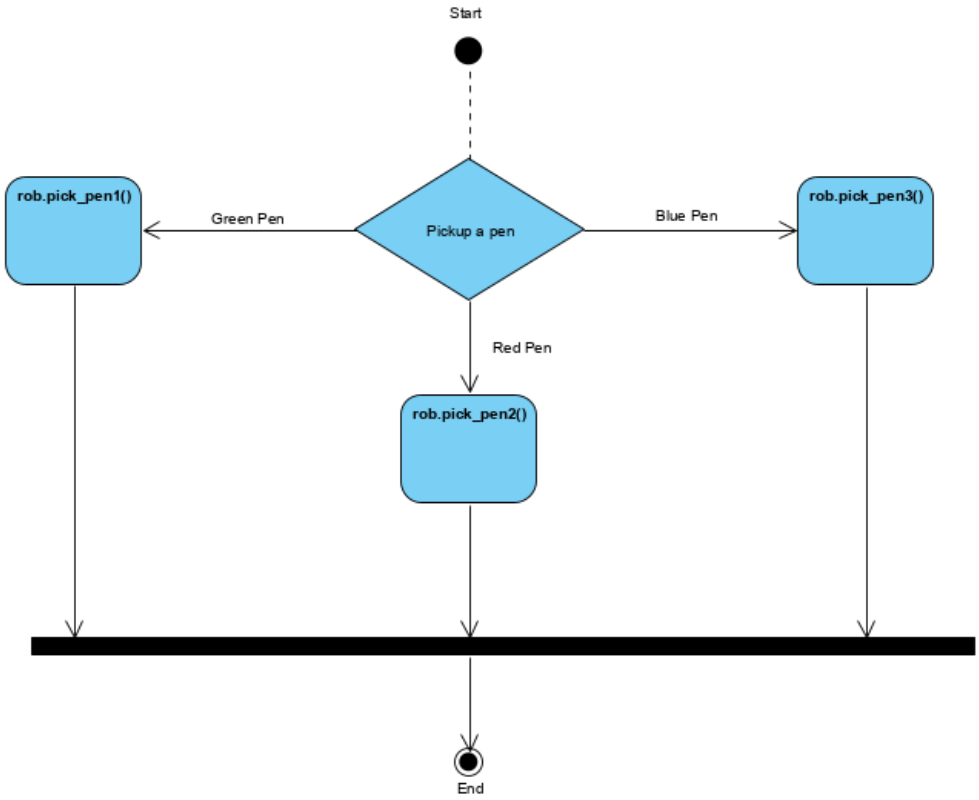


Figure 57. *Activity Diagram for Pen Pickup Decision*

For cell 8, Blue pen can be requested by using the following URL using the HTTP POST method:

1. http://192.168.8.11/rest/services/picBluePen
2. http://192.168.8.11/rest/services/Draw1

Second URL on the list is used for a for a request to draw first shape. A user can request a shape or drawing out of 9 available options. There is only one function for drawing but 9 different set of points that are passed from server to drawing function. Whenever client requests for a drawing, server first figures out that it is a ‘draw’ request, then it checks if high-power is enabled, if not, it tells the user through pendant to enable high power and try again. If the high power was enabled and draw request went through, server draws the request shape. Code for first set of drawing points or Draw1 is given below:

```
VALUE $MID($in[0],1,4) == "draw":

$drawnumber = $MID($in[0],5,1)

IF SWITCH(POWER) THEN

PDNT.NOTIFY "Info", "Drawing starts"          ;T20 notification

CASE VAL($drawnumber) OF              ;Which point array is requested
```

```
VALUE 1:
rob.run = TRUE                                  ;Robot is working
EXECUTE/c 0 rob.draw(drawpoints1[,]), 1         ;execute drawing on task 1
WRITE (lun, handle) $ENCODE("drawStarted")      ;message to client
WAIT (rob.run == FALSE)                         ;wait for drawing
WRITE (lun, handle) $ENCODE("drawEnded")        ;message to client
PDNT.NOTIFY "Info", "Drawing Ended"
END
ELSE
PDNT.NOTIFY "Info", "can't execute Draw. Enable high power and try again"      ;T20 notification
END
```

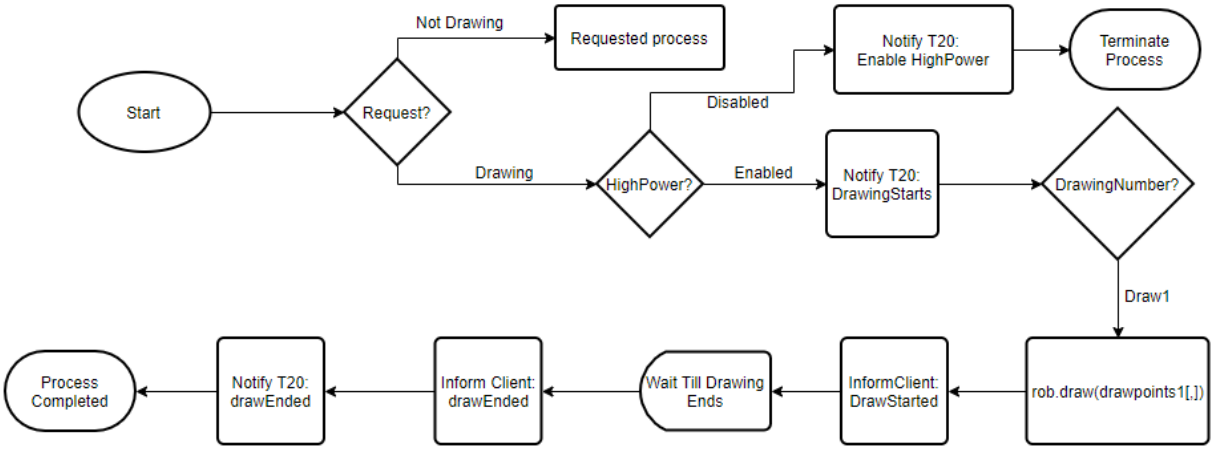


Figure 58. *Code Flow for Draw1*

In case, the draw points are sent from the python file, server identifies them, separates the coordinates and populates the draw points array with those coordinates. If server identifies that first letter or receiving message is X, it takes it as incoming array of points. After receiving the array, it removes the first letter from first index of the array which is letter 'X'. Following code line shows the message sent from python script:

```
MESSAGE = "X:"+ str(coord[0]) + " Y:" + str(coord[1])+ " Z:" + str(coord[2]) + ":" + sys.argv[2]
```

After removing the first letter 'X' from the script, it removes the ':' or from the 2nd index.

```
$temp = $DECODE($in[0],":",0) ;removes X
$temp = $DECODE($in[0]," :",1) ;remove empty space and :.characters
```

After removing ‘X’ and ‘:’ the next value in received message is the actual coordinates for x-axis, so they are stored in the relevant variable:

```
$xvalue = $DECODE($in[0]," :",0)
```

After the x-coordinates, message contains the y coordinates, but to differentiate between x and y values, a space has been placed between the x coordinates and Y character. So, the program first removes that space and then it similarly removes the ‘Y’ and ‘:’ characters.

```
$temp = $DECODE($in[0]," :",1) ;removes space
$temp = $DECODE($in[0],":",0) ; removes letter Y
$temp = $DECODE($in[0]," :",1) ;removes ‘:’
$yvalue = $DECODE($in[0]," :",0) ;saves coordinates for y-axis
```

The process to get coordinates for z-axis are same as y-axis. After all 3 coordinates have been received, server finds the draw index in the receiving message, which specifies the array number in which the coordinates will be saved. Process for differentiating the draw number from the received message is same as the process for xyz-coordinates.

```
$temp = $DECODE($in[0]," :",1) ;removes ‘:’
$drawnumber = $DECODE($in[0]," :",0) ;saves draw number
```

After receiving the draw number, server passes all the values to the srvr.addpoint function, which saves those values in the related draw point arrays.

```
CALL srvr.addpoint(VAL($xvalue), VAL($yvalue), VAL($zvalue), VAL($drawnumber))
```

After calling the function, server sends back a message saying “ok”. But before starts sending a draw points, python script makes sure that draw point array is empty, which is done by calling a clear draw points function.

```
MESSAGE = "clear" + sys.argv[2]
s.send(MESSAGE.encode('utf-8'))
```

Message contains the string “clear” to tell server which case should be processed and sys.argv[2] contains the draw point array number. Hence server first calls the srvr.clear-draw function and then srvr.addpoint function.

Functions for clearing the draw points is defined as: srvr.cleardraw(drawnumber), only argument this function receives is draw number, and it changes xyz-axis values to -1 in the array associated to that draw number. While add points function is defined as: srvr.addpoint(x, y, z, drawnumber), it received coordinate values and draw array number to add these coordinates to relevant array. It first finds the last non -1 value of x and y coordinates, and then starts populating the draw point array the from that point back-wards.

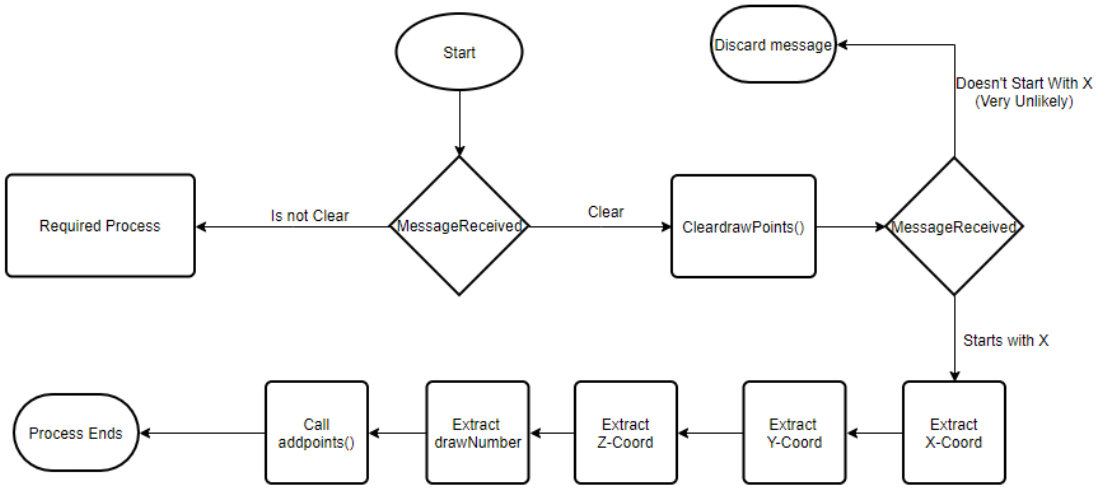


Figure 59. *Adding Points to Draw Points Array*

Configuration function is called in the same manner from the server, if there is a request to configure the drawframe, server calls the configuration function and notifies the T20 Pendant if high power is enabled otherwise, it displays an error message requesting to enable for high power and then user should try again to send the configuration request.

4.4.2 **Main**

Functions of the Main module or the program are shown in the figure 60. Close Tool and Open tool are used to activate the outputs that are associated to the pneumatic solenoid valves, which in turn close and open the gripper finger. Flow on and Flow off functions are used for the valves that operate the slider. Flow on brings the slider to most extended position, while flow off turn that valve off and cuts off air supply, making the slider flexible.

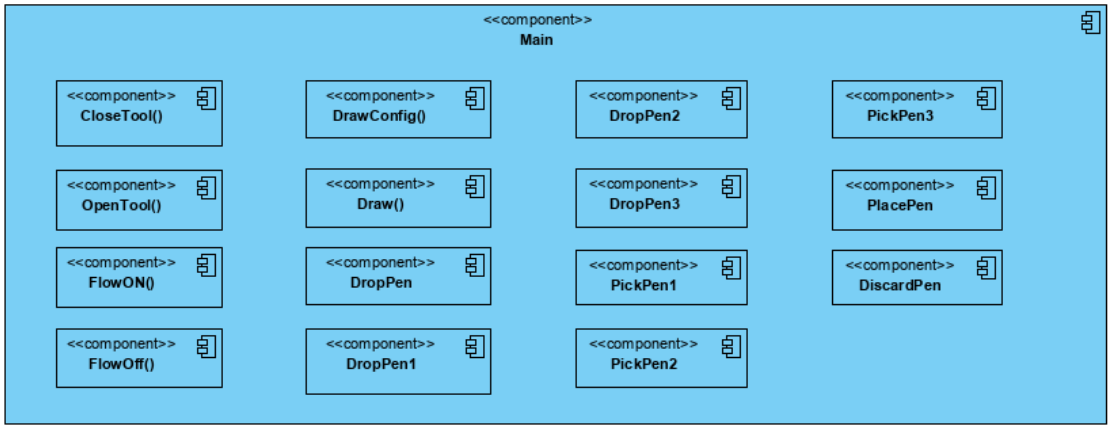


Figure 60. *Main Module's Functions*

Functions that can be called from the server are 3 pick pen function, draw and draw configuration functions. Drop pen functions are called from pick pen functions. In case, server receives a request to pick a pen, server calls the pickpen function. Pickpen function first verifies if there is already a pen in the gripper, pen presence is checked via the photoelectric sensor installed on the gripper. If there is already a pen, it calls DropPen function. After that, OpenTool function is called, which open the gripper, if it fails then program stops and notifies the pendant. Otherwise, it picks up the requested pen. After picking up a pen, it goes back to home position, see figure 61.

DropPen function figures out which pen is already present in the gripper, and it calls the function that can drop that pen back to its position.

```
IF pen_on_board == 1 THEN
    CALL rob.drop_pen1()
END
IF pen_on_board == 2 THEN
    CALL rob.drop_pen2()
END
IF pen_on_board == 3 THEN
    CALL rob.drop_pen3()
 END
```

DropPen function, drops the pen and goes to a safe location defined near the pen holder so, it can get ready to pick up the next pen. In case of failure to drop a pen, it goes back to home location since now, PickPen() function will not do anything as seen in figure 62.

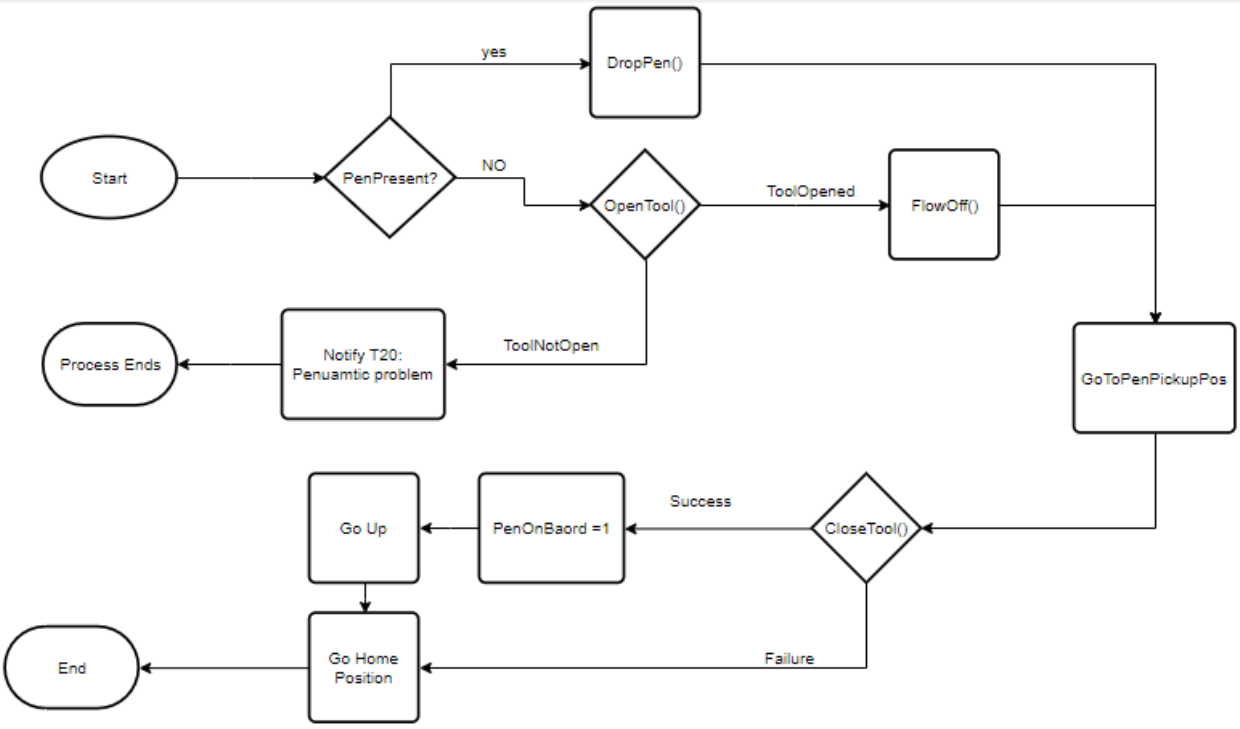


Figure 61. *Pickup Process for Pen1*

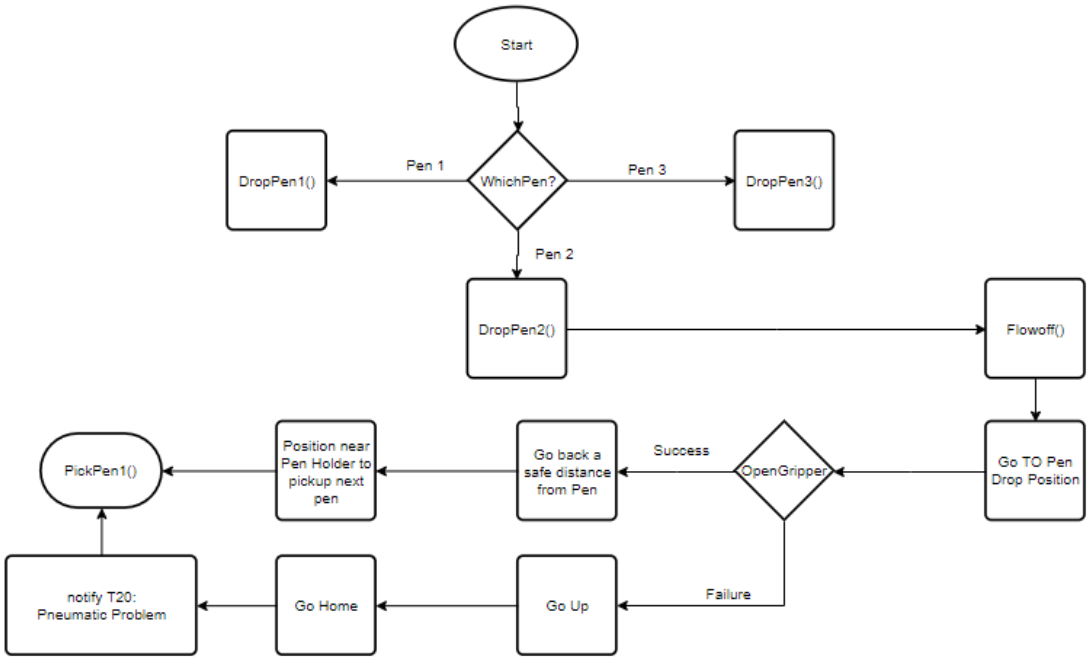


Figure 62. *DropPen2() Call from PickPen1()*

In case, Gripper does not open while dropping pen, PickPen() function will not do anything as seen in figure 62. PlacePen() and DiscardPen() functions are used to put the pen in the mentioned place, they check if gripper is holding the pen. Then they go to the appropriate drop place and put the pen there. In case of pneumatic problems, these functions show an error message.

Drawframe configuration is important part of this implementation. It is implemented by defining three points: Origin, drawx and drawy points; drawx and drawy are points to the x-axis and y-axis of the origin. These points are defined through the teach pendant, but they are updated in the configuration function, which is used to calibrate the starting position of drawing process. It is mandatory to pick up a pen before configuration, after picking up a pen, when draw_config() function is called, robot goes to the user defined origin. Here, it calls the flowOn function, and slider becomes fully extended and its sensor gets activated. Now, manipulator starts going down at origin till the slider sensor is giving a true value, as soon as, slider signal cuts off, manipulator stops, goes a bit up and updates the origin to that point.

After updating the origin, manipulator goes up and then repeats the same process at drawx and drawy coordinates. After updating all three points, program defines the draw-frame position from these points. 4^{th} value in the frame defines the orientation of the tool.

```
SET drawframe=FRAME(draworigin, drawx, drawy, draworigin)
```

Now, after defining the drawframe coordinates, robot is ready to be used for drawing. During conversion of picture to draw points, value of z-axis defines it robot should draw in that area of not. If value of z-axis is zero, that means robot should not draw, if it is 1, then robot should draw.

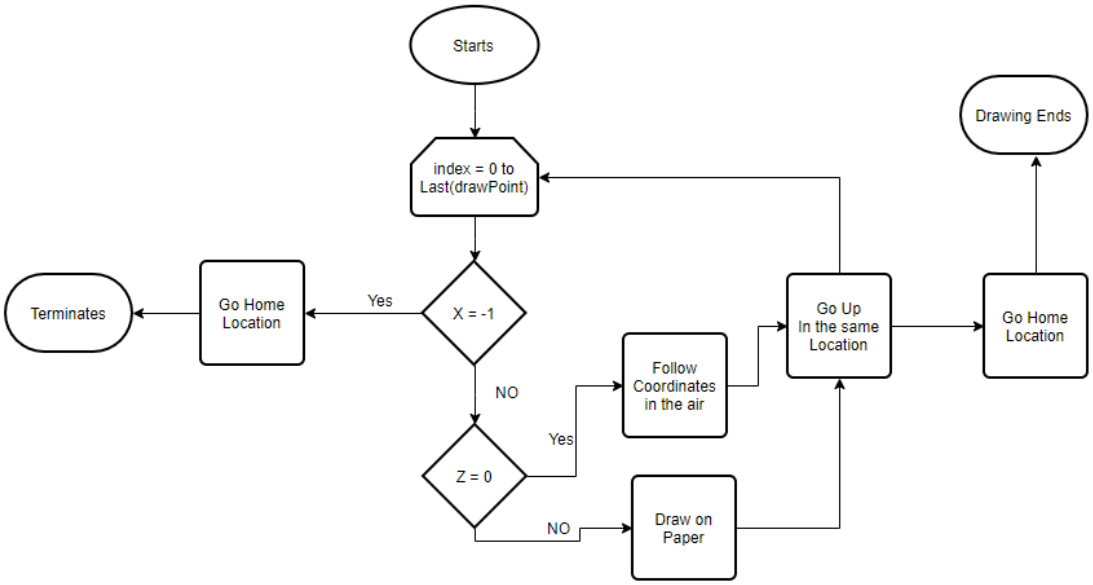


Figure 63. *Drawing Process Flow*

4.5 Free Shape Algorithm Implementation

Path planning is an important part of this implementation. For tasks like drawing, free shape algorithms can be used for manipulator's end effector movements. In the section 2.6 of this document Bezier curves and De Casteljau algorithm has been defined. This section covers them in more details and describes how Bezier curves were implemented for this thesis.

Bezier curve points can be defined by evaluating a parametric function, which is expressed in a polynomial form. [82] It was described earlier that cubic Bezier curves, or third order Bezier curves can be used to accurately follow complex paths. Cubic polynomial for third order curve is given as:

$$B(t) = (1-\tau)^3 P_0 + 3t(1-\tau)^2 P_1 + 3t^2(1-\tau)P_2 + \tau^3 P_3 \qquad (6)$$

Where τ is a function parameter and it ranges from 0 to 1 i.e. $\tau \in [0,1]$. The entire curve points can be reconstructed by evaluating the polynomial by changing function parameter from 0 to 1, its each value gives a different point on the curve. It was described earlier that curve always start at P_0 and stop at P_n, a cubic Bezier curve starts at P_0 and ends at P_3. The curve is always tangent to $\overline{P_0P_1}$, $\overline{P_1P_2}$, $\overline{P_2P_3}$ here $P_1$and P_2 are gravitational points or also known as control points, and the curve tend to go through them hence, the arrangement of these points control the shape of curve. For a cubic Bezier curve, these four points are known as descriptors.

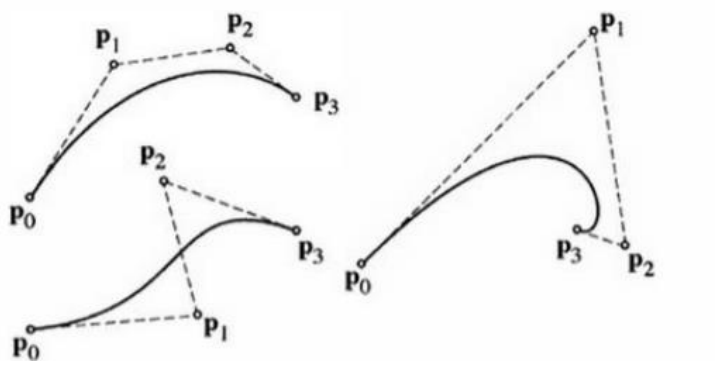


Figure 64. *Third Order Bezier Curves [82]*

Complex shapes can be approximated by using piecewise Bezier curves, which are called B-spline curves. According to Lastra, Lobov & Moctezuma, in order to make a robot controller able to interpolate Bezier either a composition of arc and linear segments is used or composition of linear segments. [82]

For the first composition, algorithm first finds the inflection points and then subdivides the curve at inflection points in biarcs with acute angles. For second composition, Bezier curve is approximated by computing and joining points with linear line segments. By increasing the number of points, better approximation can be achieved, but it increases the amount of data. Points can be computed by evaluating the equation directly or by using De Casteljau algorithm. Direct method becomes unstable for lower values of function parameter.

De Casteljau algorithm is slower but stable method, and it computes a single point of the curve by interpolating 6 compositions which are given in figure 65. In these interpolations, smaller increment step for function parameter, brings more data but closer approximation of Bezier curves.

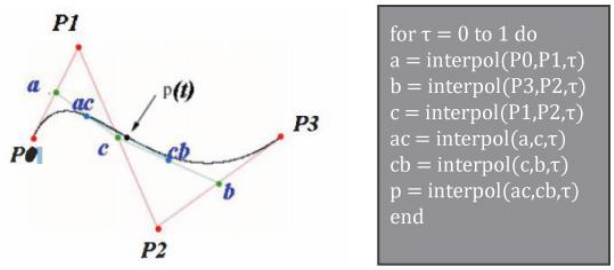


Figure 65. *Graphical Representation & Pseudocode for De Casteljau algorithm [82]*

For arc approximation method low interpolations are required hence amount of data is less as compared to linear approximation, also the movements with arc approximations are smoother relatively. But linear approximation algorithm is relatively less complex, and it can be applied for any Bezier curves while arc approximation fails in certain cases.

For this thesis, python's svgpathtools were used for point calculation. The script is used via sys.argv list from python. Sys.argv has arguments, which represent following things respectively: sys.argv[0] is name of the python script, sys.argv[1] is used for filename or picture name in this case, while sys.argv[2] is used to represent the draw points array number that ranges from 1 to 9. The picture is first converted to dot svg and then it can be used with this script. The following statement is used to call the python's path tool for conversion from svg to paths.

```
paths, attributes = svg2paths(sys.argv[1])
```

Svgpathtools contains bezier.py that can be used for nth order Bezier curves. This svg tool has five path segment classes: path, Line, Arc, Cubic Bezier and Quadratic Bezier. These classes take different parameters given as:

- Line: start and end points.
- Path: Segments
- Arc: Start point, radius, rotation, large_arc, sweep and end point.
- Quadratic Bezier: Start, Control and end points
- Cubic Bezier: Start point, control 1 and control 2 points, and end point.

The picture provided to python script is svg format, it is converted to a list of path objects and attributes of each path, as shown in program 1 of this section. These paths are made of cubic Bezier objects. The returned coordinates are in the form of complex numbers where real part is used for x-axis and imaginary part represents y-axis.

```
path:Path(CubicBezier(start=(25.327696+9.845238j), control1=(25.327696+9.845238j), control2=(19.117831+9.580988j), end=(19.117831+14.469606j)),
     CubicBezier(start=(19.117831+14.469606j), control1=(19.117831+19.358223j), control2=(24.799197+18.565475j), end=(24.799197+18.565475j)),
     CubicBezier(start=(24.799197+18.565475j), control1=(24.799197+18.565475j), control2=(19.117831+19.622473j), end=(19.514205+22.925593j)),
     CubicBezier(start=(19.514205+22.925593j), control1=(19.910579+26.228713j), control2=(18.985706+32.042204j), end=(21.496077+32.306453j)),
     CubicBezier(start=(21.496077+32.306453j), control1=(24.006448+32.570703j), control2=(26.913194+32.570703j), end=(26.913194+32.570703j)))
seg:CubicBezier(start=(25.327696+9.845238j), control1=(25.327696+9.845238j), control2=(19.117831+9.580988j), end=(19.117831+14.469606j))
```

Figure 66. *Cubic Bezier Returned by the svg2paths Tool*

It can be seen from the picture that returned path consists of multiple segments. For this implementation, these segments were taken on by one, as shown in the picture. Each segment was converted into tuplex and these tuplex were appended as coordinates:

```
tuplex = (round(seg.point(i/5).real,2),round(seg.point(i/5).imag,2),z)

coordList.append(tuplex)
```

The z-axis value is not given by the svg path tools, in this implementation, it is normally set to one, but if it is the end of the last segment of the path, z is set to zero. In the given program 2, the index (i) goes from 0 to 5, so, for every segment, a 0.2 step size is used.

After appending all the coordinates, script sends a message to clear the point array already present on that draw number and then sends those points over a TCP socket to populate the respective draw number array. Tuple of control points are used to represent Bezier curves.

```
MESSAGE = "X:"+ str(coord[0]) + " Y:" + str(coord[1])+ " Z:" + str(coord[2]) + ":" + sys.argv[2]
 s.send(MESSAGE.encode('utf-8'))
```

5. SYSTEM TESTS ON THE INDUTRIAL PILOT

This section focuses on the testing of the system implemented in the section 4. It is important to define the test cases and pre-requisites and expected results to perform a test. The following sections focus on defining and validating the test suits using two different system integration testing techniques.

5.1 Bottom-up Breadth First

Table 7. *Test Suite 1: drawing tool's unit module testing*

Test suite id	TS-1
Test case summary	This test suite is used to verify that robot can pick, place and discard a certain pen when asked.
Pre-requisites	RTU and robot-server are running, and there are pens in pen holder.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/picGreenPen http://192.168.8.11/rest/services/picRedPen http://192.168.8.11/rest/services/picBluePen http://192.168.8.11/rest/services/PlacePen http://192.168.8.11/rest/services/DiscardPen
Expected results	• If gripper is empty, it will pick new pen. • If gripper is empty, it will not execute place or discard pen requests. • If gripper is holding a pen, I will execute the place or discard pen requests. • After execution or denial, robot will inform the RTU about the process completion and RTU will publish an event.
Test Cases	a. TC-1: Pick, place and discard Green pen, in the order. b. TC-2: Discard, pick and then place Red pen. c. TC-3: Place, pick and discard Blue pen, in given order.

Table 8. *Test Suite 2: Pen change operations test*

Test Suite id	TS-2
Test case summary	The objective of this test is to verify that robot can change a pen.
Pre-requisites	RTU and robot-server are running, and there are pens in pen holder.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/picGreenPen http://192.168.8.11/rest/services/picRedPen http://192.168.8.11/rest/services/picBluePen
Expected results	• When asked, robot will pick a pen if gripper it empty. • If gripper is not empty, robot will place the pen back and pick the requested pen. • After execution or denial, robot will inform the RTU about the process completion and RTU will publish an event.
Test Cases	TC-1: Pick Green pen. TC-2: Change to Blue pen. TC-3: Change to Red pen.

Table 9. *Test Suite 3: Configure z-axis Request Testing*

Test suite id	CONFIGUREZ
Test case summary	To verify that RTU can call the z-axis calibration services.
Pre-requisites	RTU and robot-server are running, and gripper is already holding a pen.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/ConfigureZ
Expected results	That RTU will request the server to calibrate the drawframe and server will inform the RTU after it has been completed. After execution or denial, robot will inform the RTU about the process completion and RTU will publish an event.
Test Cases	TC-1: z-axis calibration request.

Table 10. *Test suite 4: Draw Operation Testing*

Test Suite id	TS-4
Test case summary	To verify that robot can draw.
Pre-requisites	RTU and Robot-server are running, gripper is holding a pen, drawframe has been configured, and drawpoints have been added.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/Draw1
Expected results	• Robot will start drawing if it is holding a pen, otherwise, it will refuse.
Test Cases	TC-1: Robot is sent “Draw1” request.

5.2 Top-down Depth First

Table 11. *Test Suite 1: Configure z-axis*

Test Suite id	TS-1
Test case summary	To verify that robot can calibrate.
Pre-requisites	RTU and Robot-server are running.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/ConfigureZ
Expected results	• Robot will calibrate the z-axis.
Test Cases	TC-1: Robot is asked to calibrate z-axis.

Table 12. *Test Suite 2: Discard Pen*

Test Suite id	TS-2
Test case summary	To verify that robot can discard a pen.
Pre-requisites	RTU and Robot-server are running.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/DiscardPen
Expected results	• Robot will discard the pen it is holding.
Test Cases	TC-1: Robot is requested to discard pen.

Table 13. *Test Suite 3: Pick Green Pen*

Test Suite id	TS-3
Test case summary	To verify that robot can pick a pen.
Pre-requisites	RTU and Robot-server are running.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/picGreenPen
Expected results	• Robot will pick Green pen
Test Cases	TC-1: Pick Green Pen

Table 14. *Test Suites 4,6,8: Draw*

Test Suite id	TS-4, 6, 8
Test case summary	To verify that robot can draw.
Pre-requisites	RTU and Robot-server are running, gripper is holding a pen.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/Draw1 http://192.168.8.11/rest/services/Draw2 http://192.168.8.11/rest/services/Draw3
Expected results	• Robot will start drawing if it is holding a pen, otherwise, it will refuse.
Test Cases	TS4: TC-1: Robot is sent “Draw1” request. TS6: TC-1: Robot is requested to draw 2nd shape. TS8: TC-1: Robo asked to draw 3rd figure.

Table 15. *Test Suites 5,7: Change Pen*

Test Suite id	TS-5, 7
Test case summary	To verify that robot can change a pen.
Pre-requisites	RTU and Robot-server are running.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/picRedPen http://192.168.8.11/rest/services/picBluePen
Expected results	• Robot will change the pen.
Test Cases	TS5: TC-1: Pick Red Pen TS7: TC-1: Pick Blue Pen

Table 16. *Test Suite 9: Place Pen*

Test Suite id	TS-9
Test case summary	To verify that robot can place a pen back to pen holder.
Pre-requisites	RTU and Robot-server are running, gripper is holding a pen.
Test procedure	Related APIs are called via the testing software.
Test data	http://192.168.8.11/rest/services/PlacePen
Expected results	• Robot will drop the pen at the right position in the pen holder.
Test Cases	TC-1: Place a pen

5.3 Test Cases Implementation

The tests are implemented by using mentioned two algorithms. In the first algorithm, drivers used are the pen numbers, provided to the system to decide where it should place the pen among pen holding slots. All other functions on the first level i.e bottom work independently. Figure 67 show the first tree that was implemented, in this tree:

- Numbers 1 to 9 are used to test basic system units individually. Testing go through all kind of condition, like Discard function, it is called thrice, but it works only one time. Other two time, there is no pen in the gripper to discard. It is used to verify that all conditions are met in individual functions.
- Number 10, 11 and 12 are not acutally called during testing. They explain the functions that are designed for drawing tools. Discard function is common for all 3 pens, and one function is used to discard any required pen.
- Number 13 and 14 are used to verify the pen change operation. It verifies the integration of pen placement and pen picking up functions. Even though only 2 combinatons are called out of all possiblities, but it still verifies that functions are integrated in the right manner.
- Z-axis callibration and only Draw1 functions are called out of 9 drawing functions to verify that point transfer alogrith and callibration functions are working in the right manner.
- The functions are called in the serial sequence, and their order is shown in the first test that was implemented. After calling a request, system waited till the RTU publishes the task complete the event.

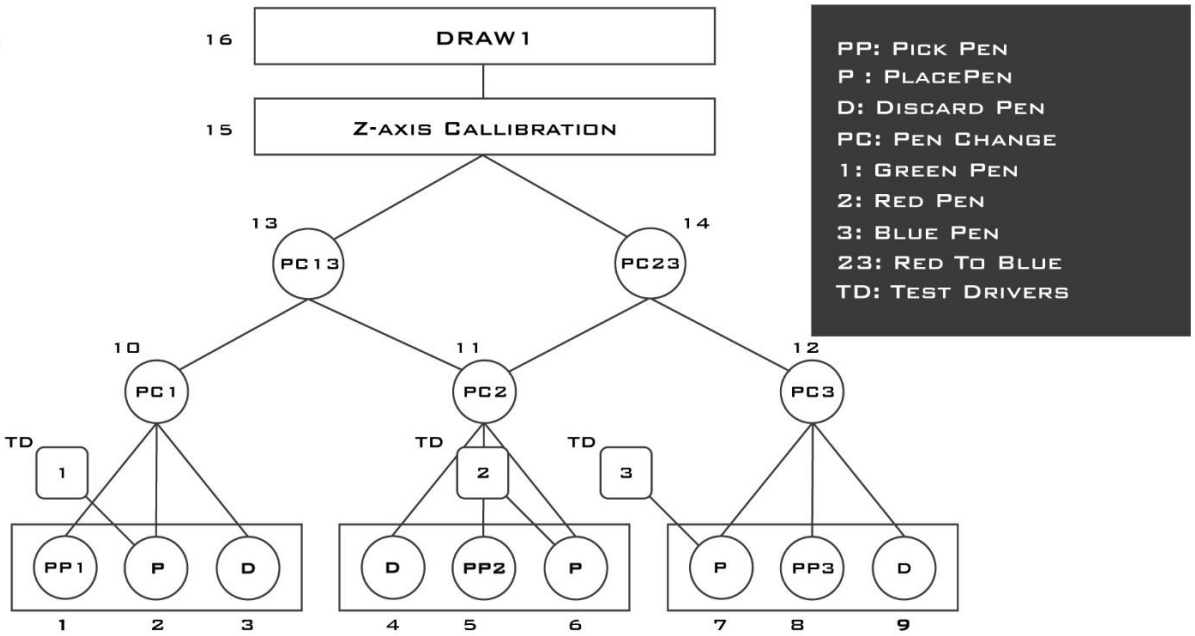


Figure 67. *Bottom-up Breadth First Tree*

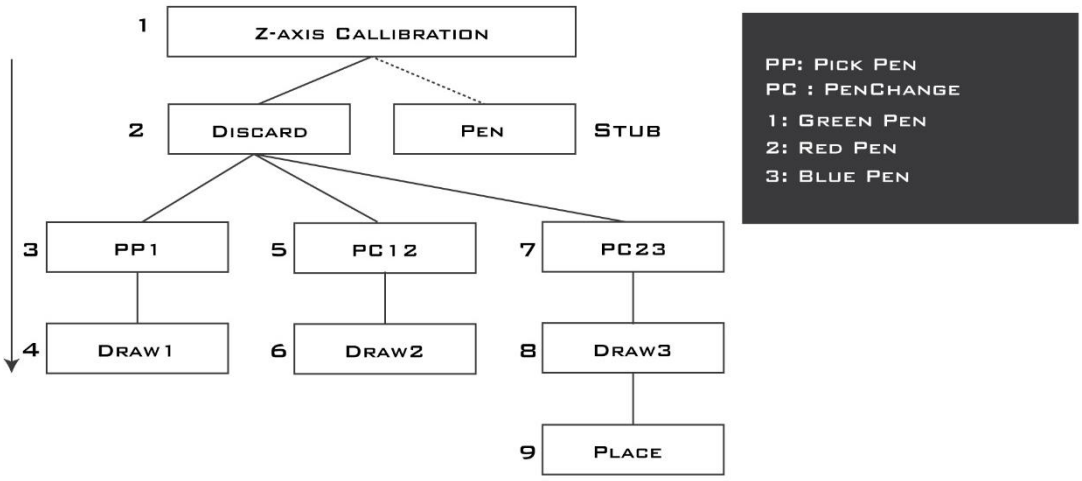


Figure 68. *Top Down Depth First Tree*

Figure 68 show the Top Down Depth First test implementation, it starts from the calibration. For calibration, gripper should be holding a pen, that is why stub is added to provide the system with required resources. Even though stubs are called from the main function under testing, but in this case, pen was provided by calling one of the pick pens functions at the start of test suite.

This test verifies the actual flow of the system, in which the robot will be working in the FASTory line. Functions are called according to the numbers mentioned in the figure.

This system also flows in the serial sequence. Katalon Studio version 6.3.3 was used for testing, and before testing, robot was already installed at the FASTory. So, the testing was done in the Targeted physical environment. Katalon studio is free, Java/Groovy based automated testing tool, and it supports number of other languages such as python.[83] The reasons for choosing this tool specifically are given below:

- It allows automated API testing or web services testing in serial and parallel manner, and it provides ready to use data functions for data evaluation. Test cases and test suits can be managed in quite organized manner with all the given test scenarios.
- Reporting system can be analysed in different ways using the Katalon Analytic tools. It keeps track of all the test cases executions and provides graphs and status reports on passed, failed and incomplete tests.

During testing, APIs are called via Katalon studio's test cases, there are 9 test cases which are used multiple times to design the test suits shown in the figures. Each test case in a way that it can be used more than once in test suites and can be re-used for other system testing as well. Each test has three main functions:

- To make a post request and verify the API response.
- To call and subscribe for a certain event and then wait for the notification to stop move towards the next test. Event subscription and listener are added in each test cases to make sure that those test cases can be reused multiple times in any order, and then they can be used for unit testing in case needed.
- Third function of the test code is to make sure that system doesn't end up waiting for ever for the event. A timer has been added in each test case, that waits for a certain time span for each test case then is closes the listener and moves towards next test case.

5.4 Test Results

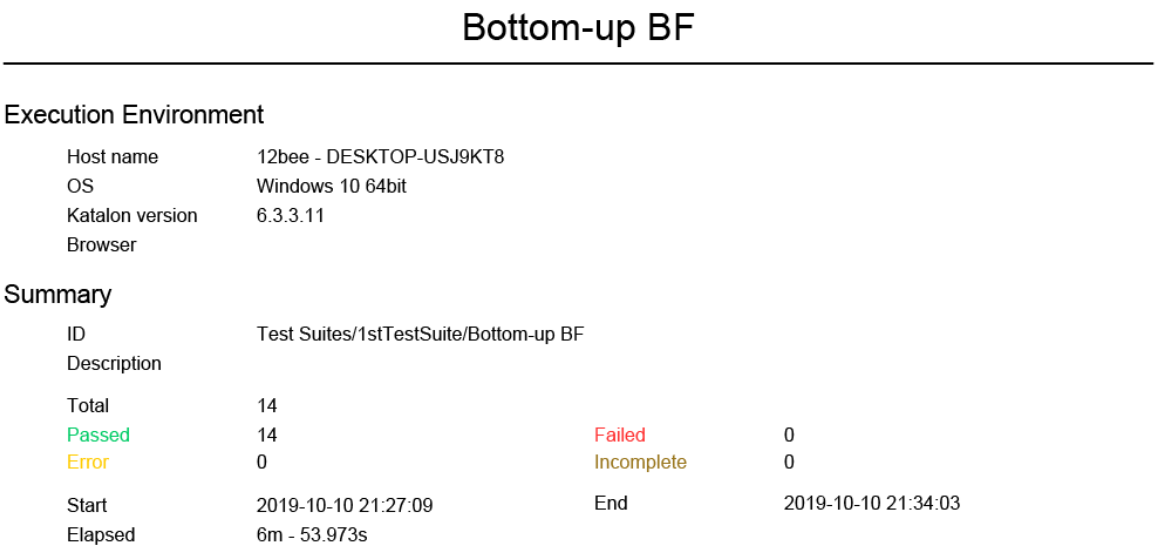
The basic purpose of testing was to verify the communication between Orchestrator, RTU and Robot. The HTTP status code of the API post request was compared against 200 status code, to verify if the RTU is performing the required actions that was requested. 200 status code stands for "OK" which means request has succeeded

Name	Total Duration	Average Duration	Request Count
Object Repository/API_Requests/BluePen	8s	4s	2
Object Repository/API_Requests/Configure	5s	5s	1
Object Repository/API_Requests/DiscardPen	5s	5s	1
Object Repository/API_Requests/Draw1	5s	5s	1
Object Repository/API_Requests/Draw2	5s	5s	1
Object Repository/API_Requests/Draw3	5s	5s	1
Object Repository/API_Requests/GreenPen	4s	4s	1
Object Repository/API_Requests/PlacePen	15s	5s	3
Object Repository/API_Requests/RedPen	4s	4s	1

Figure 69. *API Performance Table*

Figure 69 shows the API performance for different test cases, it shows the time taken by a post request during a test. On average it took 4.5 seconds to make a post request, it also shows how many times a function was called during the test. Request response was verified during the tests, following 2 figures show the different test cases that were called during the testing and their status. Figure 70 and 71 show the number of test cases called during the test, passed test cases, and time taken by each test. Time mentioned in the pictures is the time taken from starting the test to the end of last task, each testcase is called after the previous test case has been completed and marked as passed based on its published event. Even though tests are completed after the related event has published, but enough delays were added to compensate for the system lags and maximum

event wait time was also set to 42 seconds for pen related operations and configuration and 2.5 minutes for drawing.



Bottom-up BF

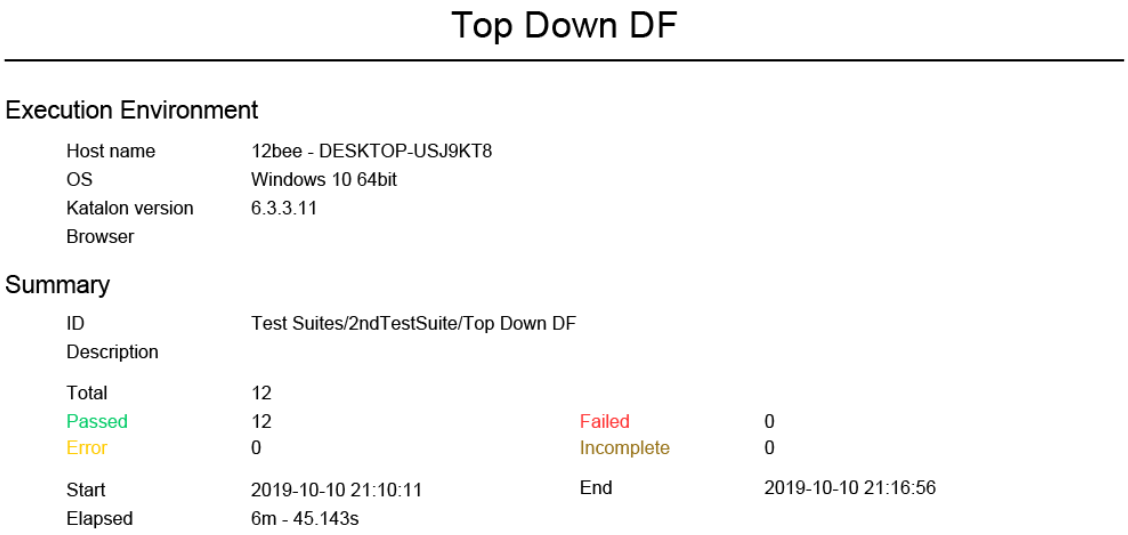
Execution Environment

Host name	12bee - DESKTOP-USJ9KT8
OS	Windows 10 64bit
Katalon version	6.3.3.11
Browser	

Summary

ID	Test Suites/1stTestSuite/Bottom-up BF		
Description			
Total	14		
Passed	14	Failed	0
Error	0	Incomplete	0
Start	2019-10-10 21:27:09	End	2019-10-10 21:34:03
Elapsed	6m - 53.973s		

Figure 70. *Bottom-up BF Test Summary Report*



Top Down DF

Execution Environment

Host name	12bee - DESKTOP-USJ9KT8
OS	Windows 10 64bit
Katalon version	6.3.3.11
Browser	

Summary

ID	Test Suites/2ndTestSuite/Top Down DF		
Description			
Total	12		
Passed	12	Failed	0
Error	0	Incomplete	0
Start	2019-10-10 21:10:11	End	2019-10-10 21:16:56
Elapsed	6m - 45.143s		

Figure 71. *Top Down DF Test Report Summary*

Figure 72 shows the graph of 14 different executions of both tests repeatedly. Some of the tests were run in loop and iterated repeatedly, API count went up to 70 times during 6th execution, making the test time 34 minutes and 70 seconds. All the API requests went through successfully and response were verified. This ensures system's ability to perform same tasks repeatedly in the assembly line.

Figure 72. *System's Execution vs Duration Graph*

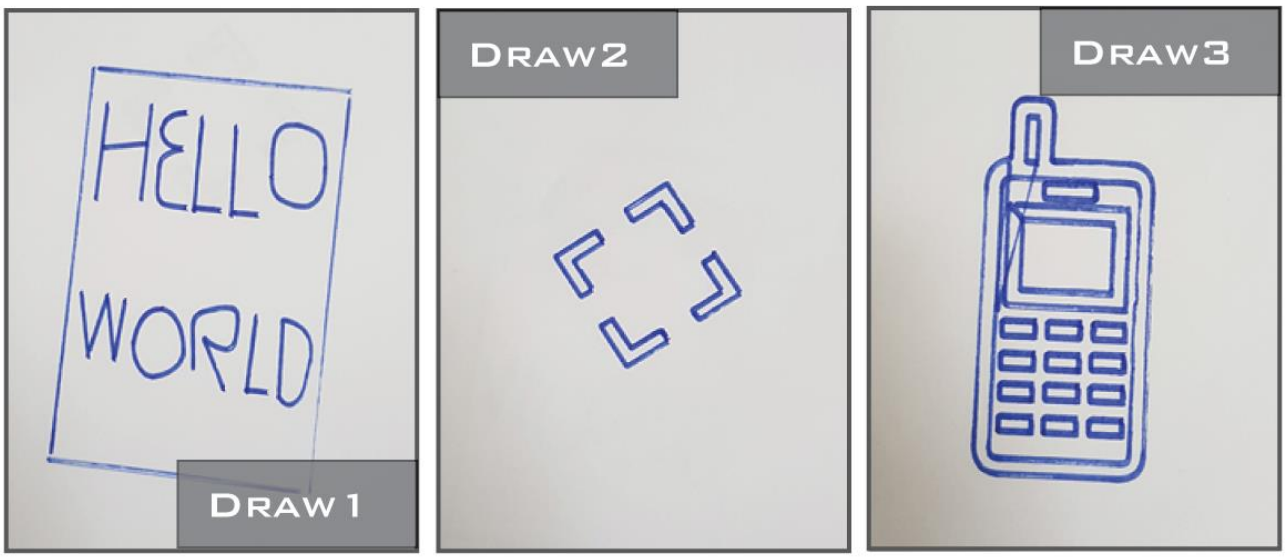


Figure 73. *Three Drawing Outputs of OMRON Robot*

CONCLUSIONS

In the problem statement, it was mentioned that system integration is a complex process because it is developed from several subsystems provided by different manufacturers. The complexity arises from the communication requirements of subsystems and identification of communication errors in the integrated system takes lots of time and efforts. This thesis provided a way to reduce the complexity by using the system integration tests to identify the errors in the integrated modules. These errors can be identified, isolated and fixed by using the proposed methodology to perform integration test in figure 33 of section 3.6.

It was further stated that automated tests have been developed to test software systems and this thesis described a way to use existing test approaches for physical industrial equipment. During this development, it was implied that the success of an integrated system lies in flawless communication between different modules and SITs can be used to test the communication between the modules using the existing software testing techniques. Automated SITs for physical systems are more time consuming, since physical system takes more time to perform a task as compared to software systems and test system should wait for the completion of task before start testing the next combined modules.

Furthermore, it was described that path planning is an important part of robotic cell deployment, to make sure that it can perform required assembly tasks while avoiding the hurdles. In this thesis, few path planning algorithms were discussed, and it was seen that Bezier curves were better option for free shape implementation. Bezier curves were implemented using De Casteljau algorithm implemented by using Python's svg to path tools library.

During implementation, path planning algorithm was not implemented in the industrial robot controller. Instead, it was implemented for a computer system, that has more computation power than an industrial controller. Points can be calculated from a given picture at any given location, there is no need to access the robot memory every time to add a new figure. Rather, it can be done more efficiently from a remote location by uploading the points to robot's memory over a network socket.

Focus of this thesis was extended to integrate webservices in a robotic cell, so they can be used to invoke and request services. Service-Oriented architecture was used to implement webservices, and robot's functionalities were implemented as services that can

be requested from an orchestrator using webservices. RTU is used as a middleware to integrate the robot controller with orchestrator.

INICO S1000 RTU is integrated with robot as a client, while robot is main service provider in the whole system. RESTful webservices are used in the implementation and services can be invoked just by posting some related APIs. Communications between robot – RTU and RTU – orchestrator are two-way communications, Services can be requested from the orchestrator, request goes to the robot via RTU, meanwhile after completing a job, robot informs the RTU about completion of the job, which publishes an event to inform the orchestrator. System is implemented using asynchronous webservices, but still next request can only be made when the previous request has been served and S1000 called back the endpoint provided during the event notification subscription.

Last objective of the thesis was to design an automated integration test to verify the system integration. Two different integration techniques (Top-Down depth first and Bottom-up breadth first) were used to verify the system. Both techniques were executed using Katalon studio and they were executed various times in different number of iterations, execution plot is shown in Figure 70. Testing was started from unit testing and moved forward testing integration of different units, and it ended at the top, verifying the entire system. Maximum of 214 API requests were made in one day during testing and all of them passed successfully and system performed requested tasks in response. Which proved that system can perform the required task in desired manner, while making two-way communication with orchestrator over ethernet using TCP/IP protocol and it can perform up to the requirements of an assembly line expectations.

REFERENCES

[1] [1] S. Wyatt, “Summary - OUTLOOK on World Robotics Report 2019 by IFR,” *IFR International Federation of Robotics*, 04-Oct-2019. [Online]. Available: https://ifr.org/ifr-press-releases/news/summary-outlook-on-world-robotics-report-2019-by-ifr. [Accessed: 14-Sep-2019].

[2] [2] G. bekey *et al.*, *Robotics: State of The Art And Future Challenges*. London : Singapore : Imperial College Press ; Distributed by World Scientific Pub. Co cop. 2008., 2008.

[3] [3] B. Gates, “A Robot in Every Home,” *Scientific American Special Editions*, no. 18, pp. 4–11, Feb-2008.

[4] [4] “ROBOT | meaning in the Cambridge English Dictionary,” *Cambridge Academic Content Dictionary*. Cambridge University Press, Dec-2008.

[5] [5] R. N. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition)*, 2nd ed. Springer US, 2010.

[6] [6] E. Garcia, M. A. Jimenez, P. G. D. Santos, and M. Armada, “The evolution of robotics research,” *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 90–103, Mar. 2007.

[7] [7] K. Dautenhahn, S. Woods, C. Kaouri, M. L. Walters, Kheng Lee Koay, and I. Werry, “What is a robot companion - friend, assistant or butler?,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1192–1197.

[8] [8] J. Boyd, “Sony Unleashes New Aibo Robot Dog,” *IEEE Spectrum: Technology, Engineering, and Science News*, 01-Nov-2017. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/home-robots/sony-advanced-aibo-robot-dog-unleashed. [Accessed: 29-May-2019].

[9] [9] J. Pransky, “Social adjustments to a robotic future,” pp. 1–10, 2004.

[10] [10] A. Sirinterlikci, A. Karaman Akgul, and O. Imamoglu, “Automation and Robotics in Processes,” in *Instrument Engineers’ Handbook: Fourth Edition*, 2011, pp. 158–168.

[11] [11] D. Nitzan and C. A. Rosen, “Programmable Industrial Automation,” *IEEE Transactions on Computers*, vol. C–25, no. 12, pp. 1259–1270, Dec. 1976.

[12] [12] T. Brogårdh, “Present and future robot control development—An industrial perspective,” *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, Jan. 2007.

[13] [13] S. Y. Nof, *Handbook of Industrial Robotics*. John Wiley & Sons, 1999.

[14] [14] T. R. Kurfess, *Robotics and Automation Handbook*, 1st ed. CRC Press, 2004.

[15] [15] J. F. Engelberger, *Robotics in Practice: Management and applications of industrial robots*, Illustrated. Springer Science & Business Media, 2012, 2012.

[16] [16] “Reusability: The Key to Making Human Life Multi-Planetary,” *SpaceX*, 10-Jun-2015. [Online]. Available: https://www.spacex.com/news/2013/03/31/reusability-key-making-human-life-multi-planetary. [Accessed: 01-Jun-2019].

[17] [17] “SRI International, Silicon Valley Robotics and NEDO Host US-Japan Robotics Conference ‘The Rise of the Robots’ | SRI International.” [Online]. Available: https://www.sri.com/newsroom/press-releases/sri-international-silicon-valley-robotics-and-nedo-host-us-japan-robotics. [Accessed: 02-Jun-2019].

[18] [18] W. L. Bargar, “Robots in Orthopaedic Surgery: Past, Present, and Future,” *Clinical Orthopaedics and Related Research*, pp. 31–36, Jul. 2007.

[19] [19] B. Singh and K. P, *Evolution of Industrial Robots and their Applications*. .

[20] [20] K. Miyoshi, R. Konomura, and K. Hori, “Entertainment Multi-rotor Robot That Realises Direct and Multimodal Interaction,” in *Proceedings of the 28th International BCS Human Computer Interaction Conference on HCI 2014 - Sand, Sea and Sky - Holiday HCI*, UK, 2014, pp. 218–221.

[21] [21] T. Arnold, M. D. Biasio, A. Fritz, and R. Leitner, “UAV-based measurement of vegetation indices for environmental monitoring,” in *2013 Seventh International Conference on Sensing Technology (ICST)*, 2013, pp. 704–707.

[22] [22] Y. Yang and Q. Cao, “A Fast Feature Points-Based Object Tracking Method for Robot Grasp,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 3, p. 170, Mar. 2013.

[23] [23] D. Leidner, W. Bejjani, A. Albu-Schaeffer, and M. Beetz, “Robotic Agents Representing, Reasoning, and Executing Wiping Tasks for Daily Household Chores,” in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, Richland, SC, 2016, pp. 1006–1014.

[24] [24] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “BigDog, the Rough-Terrain Quadruped Robot,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, Jan. 2008.

[25] [25] “ABB unveils the future of human-robot collaboration: YuMi® - Press Release,” 13-Apr-2015. [Online]. Available: http://www04.abb.com/global/seitp/seitp202.nsf/0/5869f389ad26c612c1257e26001c974c/$file/15_23+GPR+YuMi+Hannover+pr.pdf. [Accessed: 03-Jun-2019].

[26] [26] S. Bouchard, “With Two Arms and a Smile, Pi4 Workerbot Is One Happy Factory Bot - IEEE Spectrum,” *IEEE Spectrum*, 03-Feb-2011. .

[27] [27] “Types of Robots - ROBOTS: Your Guide to the World of Robotics,” *Robots Your Guide Through the World of Robotics*. [Online]. Available: https://robots.ieee.org/learn/types-of-robots/. [Accessed: 10-Jun-2019].

[28] [28] V. Kapila, “Introduction to Robotics,” New York University.

[29] [29] A. Ghosal, “MODULE 1 – INTRODUCTION: ROBOTICS: ADVANCED CONCEPTS & ANALYSIS,” Indian Institute of Science, Bangalore, India, 2010.

[30] [30] D. B. Williams, “An Introduction to Robotics.” Dr. Bob Productions, 2019.

[31] [31] F. SHAKHATREH, “The Basics of Robotics,” Mechatronics Thesis, Lahti University of Applied Sciences, Lahti, Finland, 2011.

[32] [32] “What Are The Main Types Of Robots?,” *RobotWorx*. [Online]. Available: https://www.robots.com/faq/what-are-the-main-types-of-robots. [Accessed: 10-Jun-2019].

[33] [33] E. Şahin, “Swarm Robotics: From Sources of Inspiration to Domains of Application,” in *Swarm Robotics*, 2005, vol. 3342, pp. 10–20.

[34] [34] J. M. Lastra, “Lecture 2: Robot Components (Part:1),” presented at the Class Lecture: ASE-9306 Introduction to Robotics and Automation, TUNI.

[35] [35] N. Boysen, M. Fliedner, and A. Scholl, “A classification of assembly line balancing problems,” *European Journal of Operational Research*, vol. 183, no. 2, pp. 674–693, Dec. 2007.

[36] [36] S. Ullah, Z. GUAN, J. MIRZA, and S. HUANG, “A survey on assembly lines and its types | SpringerLink,” *Higher Education Press and Springer-Verlag Berlin Heidelberg 2014*, vol. 9, no. 2, pp. 95–105, Jun. 2014.

[37] [37] G. Boothroyd, *Assembly Automation and Product Design*, 2nd ed. CRC Press, 2005.

[38] [38] K. Hughes, G. Szkilnyk, and B. Surgenor, “A System for Providing Visual Feedback of Machine Faults,” in *Enabling Manufacturing Competitiveness and Economic Sustainability*, 2012, pp. 305–309.

[39] [39] T. Feng and F. Zhang, “The Impact of Modular Assembly on Supply Chain Efficiency,” *Prod Oper Manag*, vol. 23, no. 11, pp. 1985–2001, Nov. 2014.

[40] [40] J. L. M. Lastra, “Coordinate transformations 1/2,” presented at the IRA Class Room, TUNI.

[41] [41] M. Mihelj *et al.*, *Robotics*, 2nd ed., vol. 43. Springer International Publishing.

[42] [42] J. Hing and K. Sevcik, “Breadth-First and Depth-First Search for Path Planning,” Tutorial.

[43] [43] V. Velardo, “How to Implement Breadth-First Search in Python,” *Python in Wonderland*, 18-Mar-2017. .

[44] [44] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, “Robotic Indoor Path Planning Using Dijkstra’s Algorithm with Multi-Layer Dictionaries,” in *2015 2nd International Conference on Information Science and Security (ICISS)*, 2015, pp. 1–4.

[45] [45] T.-T. Wang, X. Han, J. Zhou, and H. Chen, “Path planning for visual servoing with search algorithm,” *Advances in Mechanical Engineering*, vol. 10, no. 1, p. 1687814017750264, Jan. 2018.

[46] [46] D. Fioravanti, B. Allotta, and A. Rindi, “Image based visual servoing for robot positioning tasks,” *Meccanica*, vol. 43, no. 3, pp. 291–305, Jun. 2008.

[47] [47] J. Choi, R. E. Curry, and G. H. Elkaim, “Curvature-continuous trajectory generation with corridor constraint for autonomous ground vehicles,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 7166–7171.

[48] [48] D. F. Nugroho, A. Dharmawan, M. I. Fikri, I. N. Ahmad, and M. R. Fuadin, “Implementation of De Casteljau’s Algorithm for Pattern Generation on BIOLOID Based Humanoid Robot’s Leg Movement,” in *ResearchGate*, Yogyakarta, Indonesia, 2018, p. 4.

[49] [49] A. Kotani and S. Tellex, “Teaching Robots to Draw,” presented at the International Conference on Robotics and Automation, Montreal, Canada, 2019, pp. 1–7.

[50] [50] M. C. Lau, J. Baltes, J. Anderson, and S. Durocher, “A portrait drawing robot using a geometric graph approach: Furthest Neighbour Theta-graphs,” in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2012, pp. 75–79.

[51] [51] Yan Lu, J. H. M. Lam, and Y. Yam, “Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator,” in *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2009, pp. 578–583.

[52] [52] S. Calinon, J. Epiney, and A. Billard, “A humanoid robot drawing human portraits,” in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 2005, pp. 161–166.

[53] [53] N. Tillmann, W. Grieskamp, and W. Schulte, “Unit test generalization,” US 7.587,636 B2, Sep-2009.

[54] [54] “Unit Testing Tutorial: What is, Types, Tools, EXAMPLE,” *Guru 99*. [Online]. Available: https://www.guru99.com/unit-testing-guide.html. [Accessed: 11-Sep-2019].

[55] [55] S. Kolodiy, “Unit Tests, How to Write Testable Code and Why it Matters,” *Toptal Engineering Blog*. [Online]. Available: https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters. [Accessed: 11-Sep-2019].

[56] [56] S. Elbaum, H. N. Chin, M. B. Dwyer, and J. Dokulil, “Carving differential unit test cases from system test cases,” in *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering - SIGSOFT ’06/FSE-14*, Portland, Oregon, USA, 2006, p. 253.

[57] [57] S. Bechtold, S. Brannen, J. Link, M. Merdes, M. Philipp, and C. Stein, “JUnit 5 User Guide.” .

[58] [58] V. Roubtsov, “EMMA User Guide,” *EMMA*. [Online]. Available: http://emma.sourceforge.net/userguide/userguide.html. [Accessed: 11-Sep-2019].

[59] [59] C. Poole, R. Prouse, and S. Busoli, “NUnit.org,” *NUnit*. [Online]. Available: https://nunit.org/. [Accessed: 11-Sep-2019].

[60] [60] S. Bergmann, “PHPUnit – The PHP Testing Framework,” *PHPUnit*. [Online]. Available: https://phpunit.de/. [Accessed: 11-Sep-2019].

[61] [61] “What is System Integration Testing (SIT) with Example,” *Guru 99*, 05-Sep-2019. [Online]. Available: https://www.guru99.com/system-integration-testing.html. [Accessed: 05-Sep-2019].

[62] [62] “Integration System Test,” Critical Systems Testing, Inc., Case Study, 2010.

[63] [63] “What is System Integration Testing (SIT): Learn with Examples,” *Software Testing help*, 21-Aug-2019. [Online]. Available: https://www.softwaretestinghelp.com/system-integration-testing/. [Accessed: 06-Sep-2019].

[64] [64] “Incremental Testing,” *Tutorial Point - Simple Easy Learning*. [Online]. Available: https://www.tutorialspoint.com/software_testing_dictionary/incremental_testing.htm#. [Accessed: 09-Sep-2019].

[65] [65] R. Paul, “End-to-end integration testing,” in *Proceedings Second Asia-Pacific Conference on Quality Software*, Hong Kong, China, 2001, pp. 211–220.

[66] [66] W. E. Howden, “Functional Program Testing,” *IIEEE Trans. Software Eng.*, vol. SE-6, no. 2, pp. 162–169, Mar. 1980.

[67] [67] H. K. N. Leung and L. White, “Insights into regression testing (software testing),” in *Proceedings. Conference on Software Maintenance - 1989*, 1989, pp. 60–69.

[68] [68] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[69] [69] “MES Explained: A High Level Vision,” in *White Paper Number 6*, 1997, p. 24.

[70] [70] D. Diep, P. Massotte, and A. Meimouni, “A distributed manufacturing execution system implemented with agents: the PABADIS model,” presented at the IEEE International Conference on Industrial Informatics, 2003. INDIN 2003. Proceedings., Banff, Alberta, Canada, Canada, 2003, pp. 301–306.

[71] [71] K. A. Germany Frankenthal, “Field level,” *KSB*. [Online]. Available: https://www.ksb.com/centrifugal-pump-lexicon/. [Accessed: 12-Sep-2019].

[72] [72] L. Canché, M. de J. Ramírez, G. Jiménez, and A. Molina, “Manufacturing Execution Systems (MES) Based on Web Services Technology,” *IFAC Proceedings Volumes*, vol. 37, no. 5, pp. 135–140, Jun. 2004.

[73] [73] A. Lobov, “III Lecture 02: Information retrieval: (Networked) Devices at the factory,” presented at the Introduction to Industrial Informatics, TUNI, 09-Apr-2017.

[74] [74] “Web Services Glossary,” *W3C*, 02-Nov-2004. [Online]. Available: https://www.w3.org/TR/ws-gloss/. [Accessed: 12-Sep-2019].

[75] [75] W. Qifeng and W. Zhangjian, “Web Services-based System Integration Approach for Manufacturing Execution System,” in *2011 International Conference on Internet Computing and Information Services*, 2011, pp. 469–472.

[76] [76] F. Jammes, H. Smit, J. L. M. Lastra, and I. M. Delamer, “Orchestration of service-oriented manufacturing processes,” in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, 2005, vol. 1, pp. 8 pp. – 624.

[77] [77] B. A. Forouzan, *TCP/IP Protocol Suite*, Fourth Edition. Raghothaman Srinivasan, 2009.

[78] [78] G. Torres, “How TCP/IP Protocol Works - Part 1,” *Hardware Secrets*, 28-Mar-2012. .

[79] [79] W. M. Mohammed, A. Lobov, B. R. Ferrer, S. Iarovyi, and J. L. M. Lastra, “A web-based simulator for a discrete manufacturing system,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, Italy, 2016, pp. 6583–6589.

[80] [80] “eCobra 600 Lite / Standard / Pro SCARA Robots/Dimensions | OMRON Industrial Automation,” *OMRON - Industrial Automation*, 04-Jan-2016. [Online]. Available: http://www.ia.omron.com/products/family/3516/dimension.html. [Accessed: 20-Sep-2019].

[81] [81] “Robots KR 3 AGILUS Specification.” KUKA, 28-Feb-2019.

[82] [82] L. E. G. Moctezuma, A. Lobov, and J. L. M. Lastra, “Free shape paths in industrial robots,” in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 3739–3743.

[83] [83] “A Comparison of Automated Testing Tools,” *Katalon Studio*, 24-Jan-2017.

APPENDIX A: USING TEXT STYLES IN MS WORD