UNIVERSITY OF MORATUWA

Faculty of Engineering

Registered Module No: EN3992

INDUSTRIAL TRAINING REPORT

Lanka Electronics Pvt. Ltd

From: 24/06/2019 To 06/12/2019

Date of Submission:

03/01/2020

R.M.I.P. Rajapakshe 160505J

Department of: Electronic and Telecommunication Engineering

PREFACE

This report will act as an elaboration on my six month training experience at the Lanka Electronics, while also providing information with regards to the establishment itself. The report also contains details on the projects I undertook and the role I had to play as a trainee.

Chapter 1:

This chapter will be dedicated to the history of the company, its growth and its current position. It will also focus on the company mission and ethics.

Chapter 2:

Chapter two illustrates my training experience during the six month period. It includes the details of the projects I was assigned, how I handled the issues faced during the projects and how I utilized the company resources to solve them. The chapter also elaborates on the machinery that I got to work with, and how I was trained to use them. The projects that I worked on mainly focused on areas regarding robotics, software development and electronics designing. I was also given exposure on how electronics can be utilized in horticulture with a number of field visits.

Chapter 3:

The final chapter will contain a brief summary of my experience gained at the company, the skills and knowledge I acquired and the challenges faced during the six months. It will also include a self assessment of what I have to focus more on my final year to aspire to be a better engineer. It will further elaborate on the support given by the company to complete my training program and the ways in which I see fit to improve the training experience further.

ACKNOWLEDGEMENT

I would like to take this opportunity to thank all the individuals who helped me in completing my training period of six months fruitfully.
First and foremost I am grateful to the Department of Electronics and Telecommunication Engineering, the Industrial Training Division of University of Moratuwa and the National Apprenticeship Industrial Training Authority (NAITA) for allowing me to partake in this exciting endeavour.
My sincere gratitude goes to our Head of Department, Dr. N.W.N Dayananda, training coordinator of the Electronic and Telecommunication Department, Dr. Peshala G. Jayasekara and all other lectures of our Department for being of support during the six month period.
A thank you goes to Eng. N.A. Wijewickrema, Director of Industrial Training Division of University of Moratuwa for giving us briefing on Internship, its importance, safety procedures and expected outcomes prior to internship and all other guidance and support during the training period.
Most importantly I am grateful to Mrs. Jayasinghe, CEO of Lanka Electronic (Pvt) Ltd, for giving me the opportunity to join as an intern at her company. I'm also obliged to thank Eng. Lakni Jayasinghe, my supervisor, for her valuable time and effort put in to making the internship successful. My thankfulness is further extended to Mr. Janka Kulathunga (Manager of Lanka Electronics research and development section), Mr. Dumindu Eranda and Mr. Aloka Perera for their support.

CONTENTS

CHAPTER 1

1.Introduction to the organization

1.1 Company profile

Lanka Electronics (Pvt) Ltd was founded in 1993 by a group of undergraduates of University of Moratuwa. The company manufactures consumer electronics as well as professional electronics. Lanka Electronics has been a professional electronics supplier to the government and multinational companies like Vario systems.

1.2 History of the company

The company was first founded in 1993 by a group of graduates from the University of Moratuwa including five shareholders. The first factory was built in Minuwangoda in the same year with their first product being laboratory training panels. Following that, the company started to produce TV antennas in 1996 as a consumer electronic product. EDM machines were also produced as a professional electronic product. Currently the company has two functional factories situated in Minuwangoda and Anuradhapura. It also has a research center located in Minuwangoda.

1.3 Products of the company

Both consumer electronics and professional electronics are produced in the company. The consumer electronics are produced on daily production lines while professional electronics are produced on demand.

1.3.1 Professional Electronic Products

These products were produced on demand only. All these products were produced to state of the art quality using local technologies.

1.3.1.1 Laboratory Training Panels

The first product of the company, laboratory training panels, fall into this category. Panels were produced right following the formation of the company in 1993. Their production was discontinued in 2010.

1.3.1.2 Die-Sink EDM Machine

In 1996 they started manufacturing Die-Sink EDM machines. This was a milestone moment as it was the first locally produced EDM machine in the country.

1.3.1.3 Motor Traffic Control Lights

Following the success with the EDM machines, in 1997, the company started to manufacture traffic lights for Sri Lankan government. These traffic lights were the first locally built systems and could be priced ten times less than the imported systems.

1.3.1.5 Wire Winding Machine

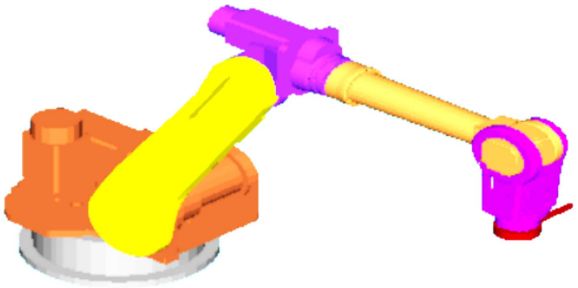
Adhering to a request made by Vario systems, in 2009, wire winding machines were manufactured. They were used to produce cable harnesses.

1.3.1.4 Electric Motor Bike

Following the trend of firsts, the first electric bike to be made in the country was also produced by Lanka Electronics in 2010. The machine was developed until one full charge enabled a 50km ride.

1.3.1.6 6 DOF Robot Arm

The 6 DOF robot arm is the latest professional piece of electronics produced in the company. It was specially designed to be useful to SME applications while being of an affordable price.

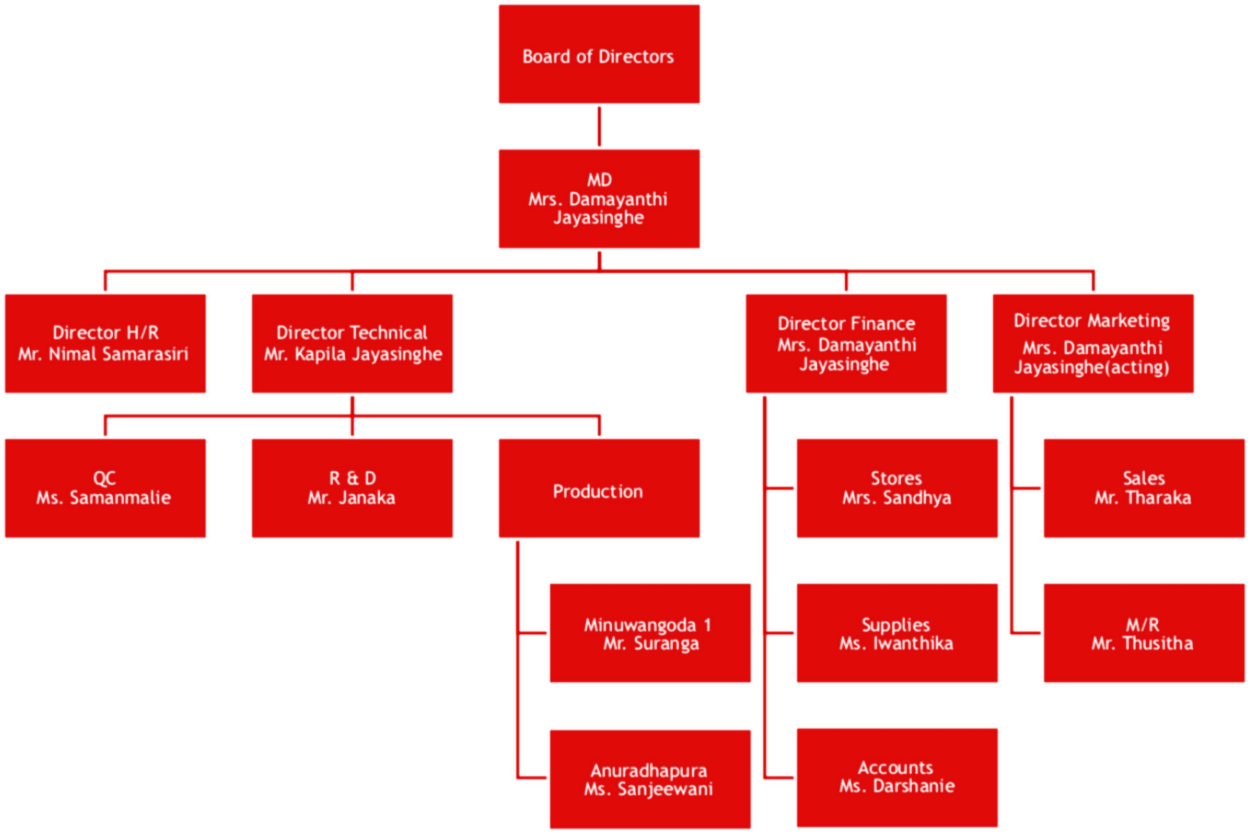


1.3.2 Consumer Electronics

TV antennas and boosters are the main consumer electronics produced in the company.

1.4 Organization Structure

There are two factories under the company and a research and development center. The factories are located in Minuwangoda and in Anuradhapura. The research center is located in Minuwangoda. The main office and stores can be found at No.7A, Vijaya Mawathe, Medemulla, Minuwangoda.
Lanka Electronics (Pvt) Ltd is governed by three board of directors and currently has a workforce exceeding 60.



1.5 Performance Analysis of Lanka Electronic (Pvt) Ltd

1.5.1 Overall Performance and Profitability

Although the company is known to produce both professional and consumer electronics, given the fact that professional electronics are only produced on direct demand, the main income to the company is from consumer electronics. The average income of the company is about 3 million per month. Around 2000 antennas are produced monthly which are distributed by the companies distribution service. The initial production of consumer electronics happen inAnuradhapura and the goods are sent back to Minuwangoda to be quality checked and stored. Raw material for production are bought on a three month basis and stored in Minuwangoda. The acquiring of raw material is estimated depending on last terms sales. Then they are forwarded to Anuradhapura on a weekly basis.

1.5.2 Usefulness to the Country

Lanka Electronic (Pvt) Ltd is a local company which has none whatsoever foreign affiliations. Local technologies are used and commendable electronics are manufactured in the company. This company is responsible for curtailing the currency outflow from the country via its consumer electronics, while ensuring an inflow by manufacturing professional electronics to multinational companies. In its own way, the company creates a number of job opportunities for locals while being an example for newly formed companies on how to function in order to create worthwhile profit.

1.5.3 SWOT analysis

CHAPTER 2

2.Training Experience

2.1 Introduction

The training I gained from my internship was mainly based on electronics design, mathematics and software development. I was a beginner in software development when I started training but now I have improved tremendously. The electronics design exposure was also enlightening and helped me improve a number of flaws in my designing. I was also assigned a project through which I acquired thorough knowledge in robotics. The training projects were assigned to pairs of trainees, and we were assigned a number of projects. My training happened at the research center of the company at Minuwangoda.

The first project we were assigned to develop a software to actuate and simulate the movement of the 6 DOF robot arm produced in the company. A path planner software was partially written before our arrival and we had to first understand the basics it ran on, and then proceed with our task. The final goal was to design a path planner capable of actuating movement in the robot such that the end effector will move with constant velocity throughout the path.

Then I was assigned a project to build a protocol interchanger for the robot end effector camera. The camera to be used in the arm is a USB camera, but due to the complexity of the USB protocol, this results in a lag when communicating with the raspberry PI. The raspberry PI was used to detect and avoid collisions and to fine tune the positioning of the robot arm by using the feedback of the camera. Hence my project was focused on building a simple device that will grab the USB protocol data from the camera and convert it to a more simplified function specific protocol such that the lag is reduced.

The third project given to us was to build a low cost Human Machine Interface(HMI). The HMI used by the robot arm currently is over Rs.40,000 and we were able to build an HMI for less than Rs.6,000. One drawback in our HMI is that the refresh rate is noticeable to the human eye and hence is not as aesthetically pleasing as the expensive one.

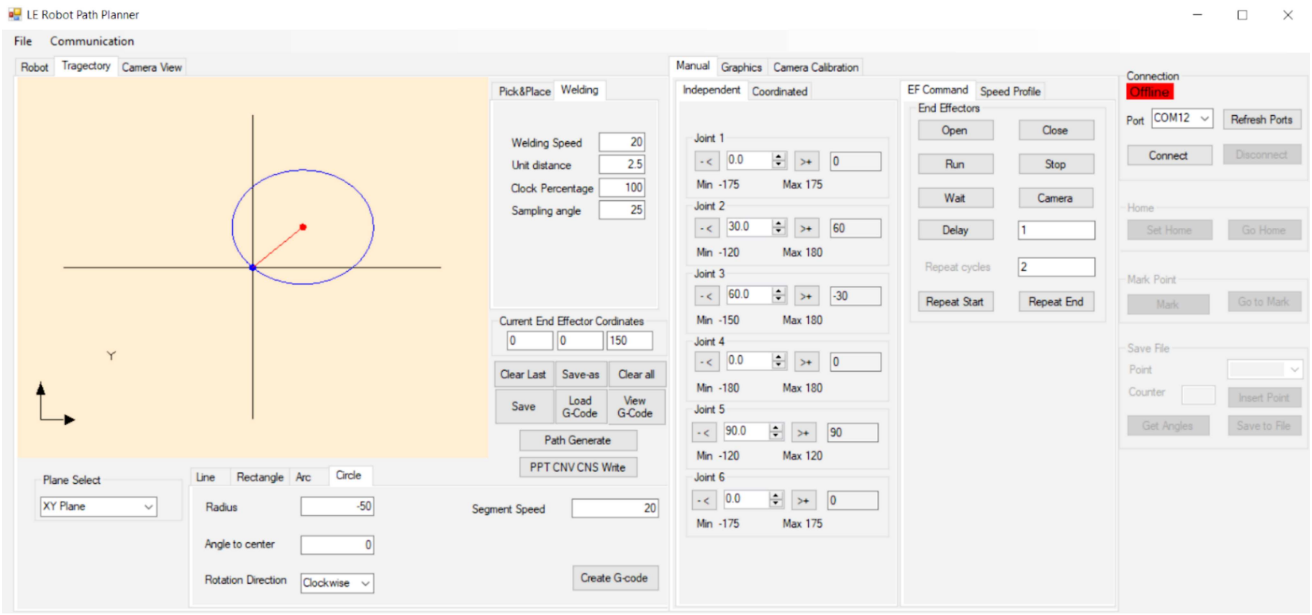
2.2 Phases of Training

2.2.1 Introductory Sessions

The team at the research center was introduced to the trainees initially and we were assigned projects as pairs on the first day. Our first project was to develop the robot path planner software. We were given a brief introduction of how a robot arm works, what areas to study and we were introduced thoroughly on the existing version of the software. We were also given a rough idea about the projects that will be assigned to us in the future and were told to be thorough with the required knowledge beforehand.

2.2.2 Robot Path Planner

The robot path planner was a software that was developed to create paths to which the robot can move. It must be understood that the robot arm works by rotating motors which in turn work by relevant voltages, which in turn are found by jerks(as in the case of our 6 DOF robot). A jerk can be introduced as the time derivative of acceleration. Already the path planner was able to calculate a list of jerks in order to move the robot arm from point to point(ppt) and also via a set of points by bypassing the middle points to create an optimum continuous path(cnt). Our task was to create a list of jerks for a path consisting of basic shapes like lines, rectangles, arcs and circles such that the end effector travels on the exact given path with an exact specified velocity.

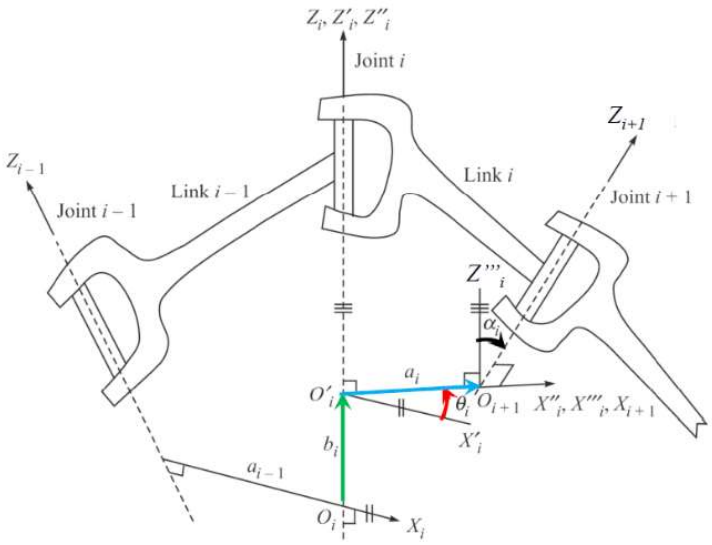


2.2.2.1 D-H Parameters

The Denavit-Hartenberg parameters are a set of four measurements that are used to describe the positioning of a link in a kinematic chain. The four parameters are

1. Joint offset -b
2. Joint angle -theta
3. Link length -a
4. Twist angle -alpha

A set of D-H parameters can successfully describe a robot arm and its related movements.



A set of D-H parameters of our robot is used to animate it in the software. A few parameters can be shown as below,

```
<RAJoint xsi:type="RARevoluteJoint">
  <JointLength>430.72</JointLength>
  <TwistAngle>180</TwistAngle>
  <JVInitial>0</JVInitial>
  <JVFinal>60</JVFinal>
  <Index>1</Index>
  <JointOffset>104.28</JointOffset>
</RAJoint>
<RAJoint xsi:type="RARevoluteJoint">
  <JointLength>0</JointLength>
  <TwistAngle>-90</TwistAngle>
  <JVInitial>-90</JVInitial>
  <JVFinal>-150</JVFinal>
  <Index>2</Index>
  <JointOffset>39.95</JointOffset>
</RAJoint>
```

2.2.2.2 Forward and inverse Kinematics

Before moving on to kinematics, getting to know about transformation matrices were important. Using the D-H parameters in the form of a matrix, the transformation matrix of a link can be found.

The translations along the D-H parameters can be shown in matrix form as follows,

$$\mathbf{T}_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T}_a = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_\theta = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T}_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where C and S stands for cos and sin.
Using the above the transformation matrix can be found as,

$$\mathbf{T}_i = \mathbf{T}_b \mathbf{T}_\theta \mathbf{T}_a \mathbf{T}_\alpha$$

Using this transformation matrix, the relevant rotation matrix and the position matrix of the link can be found respectively.

$$\mathbf{T}_i = \begin{bmatrix} \text{Rotation Matrix} & \text{Position} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Moving on to the kinematics, forward kinematics is the process of finding the end effector position depending on the D-H parameters of each joint in the robot arm while inverse kinematics, as the word itself implies, deals with giving suitable angular rotations according to the end effector position.

Consider a three link arm with the following parameters,

Link	b_i	θ_i	a_i	α_i
1	0	θ_1 (JV)	0	$-\pi/2$
2	0	θ_2 (JV)	a_2	0
3	0	θ_3 (JV)	a_3	0

The respective transformation matrices will be as follows,

Link 1

$$\mathbf{T}_1 \equiv \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link 2

$$\mathbf{T}_2 \equiv \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link 3

$$\mathbf{T}_3 \equiv \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Upon multiplication, the final transformation matrix of the arm can be received as,

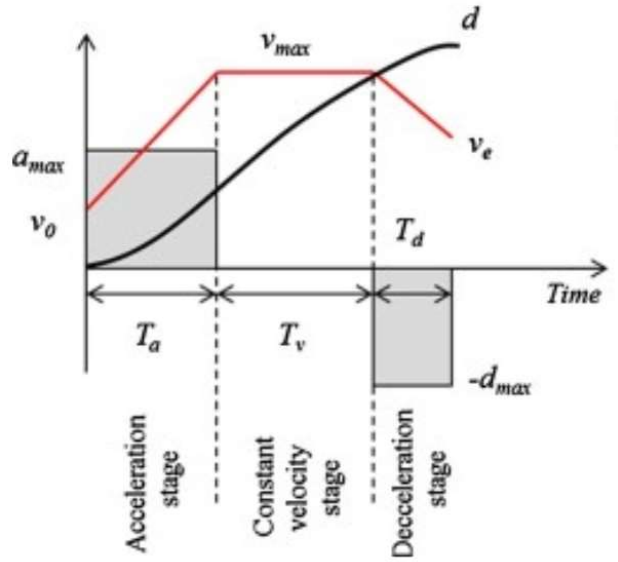
$$\mathbf{T} \equiv \begin{bmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(a_2c_2 + a_3c_{23}) \\ s_1c_{23} & -s_1s_{23} & c_1 & s_1(a_2c_2 + a_3c_{23}) \\ -s_{23} & -c_{23} & 0 & -(a_2s_2 + a_3s_{23}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using which we can determine the position of the end effector by looking at the last column of the matrix.

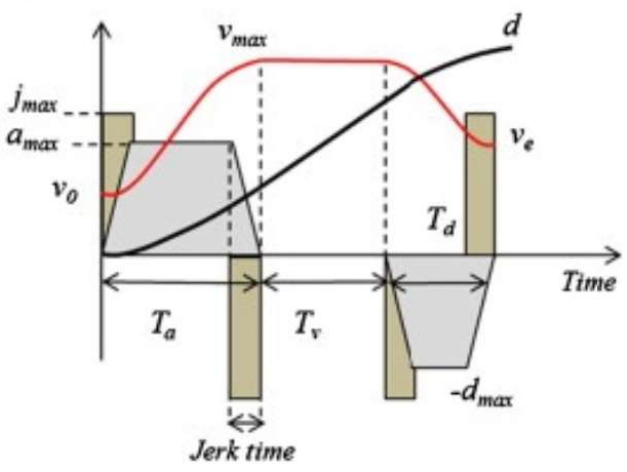
A similar but a little more complex procedure is followed in inverse kinematics.

2.2.2.3 Jerk and Motor Control

As mentioned earlier, the motors in our robot can be controlled by varying the jerk given to them. But as with any animated object a sudden change in jerk can cause a motor to slip, as it will start accelerating with a constant increase. This behaviour can be shown as follows with infinite jerks at the start and end of acceleration and deceleration stages.



But, the motors in our 6 DOF robot does not have feedback, hence a slip in the motor may cause a number of adverse effects. So in order to prevent this situation, the jerk profile given to the motor will be created such that the subsequent acceleration behaves in a trapezoidal shape, and the velocity behaves like an ‘s’, giving it the term ‘s-curve’.

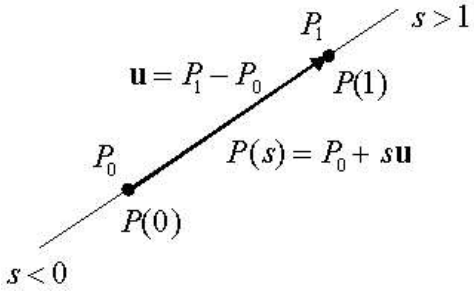


2.2.2.4 Path Mapping

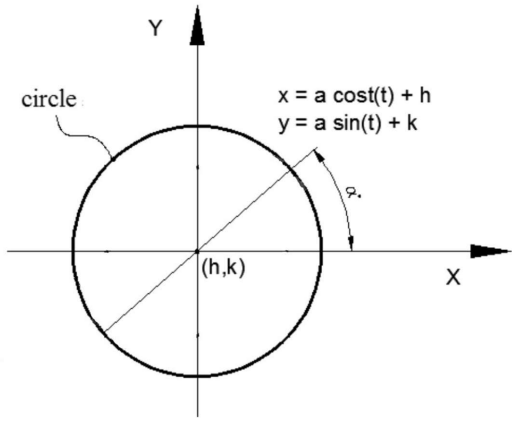
We were initially asked to make sure that the end effector of the robot moved in the exact path defined by the user without any deviations. A simplified version of our proposed solution will be as follows,

1. Breakdown the intended path to small fragments such that a variation of velocity within the fragment is negligible.
2. Solve for the respective angles of the six motors such that the placement of the end effector at the fragment ends are possible.
3. Solve for the required jerks such that the end effector velocity is kept constant.

Path mapping was the first job on hand, and to do it we tried a number of methods. First we explored on the Bresenham algorithm to map a line. This was not the best approach to our problem as we later understood that using Bresenham's algorithm will assure that our line is not exactly equal to the intended result. So we went with a basic parametrized line with the parameter being calculated based on the constant length of a single fragment.



A similar approach was taken with the arc, a parameterized circle was used, in opposition to using bezier curves, which will not lead to a perfect circular arc.



2.2.2.5 Applying Kinematics

Now that the path is mapped into sections we had to find the angles of each motor for each point in the path. For this inverse kinematics was used. As the robot has six degrees of freedom along with six motors, this could be easily done by a few matrix calculations.

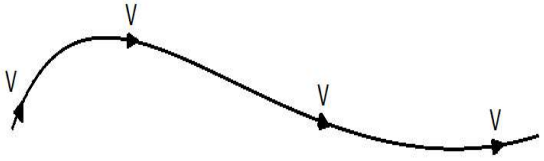
2.2.2.6 Solving for Jerks

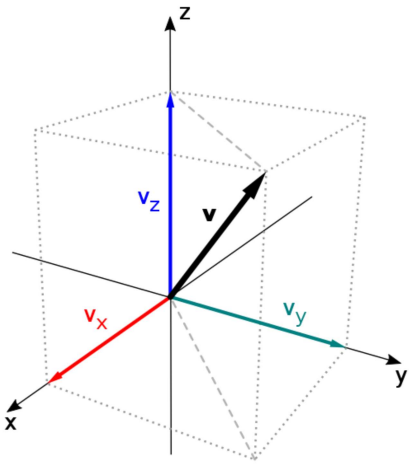
The equation below was useful in this state to calculate the jerk profiles.

$$\dot{\theta} = \mathbf{J}^{-1}[\mathbf{V} \quad \Omega]^T$$

Theta dot -Angular velocities of the 6 motors
J -Jacobian matrix
V -Linear velocity of the ned effector
PI - Angular velocity of the end effector

J, the jacobian matrix represents the rotation and the position of the end effector with respect to the base. For our initial calculation, we considered a situation where the end effector had no angular velocity with respect to the link base, hence making PI a zero matrix. The speed along the path had to be kept constant as follows.





V matrix represented by the components of a velocity point towards the basis directions. So V would look like the following,

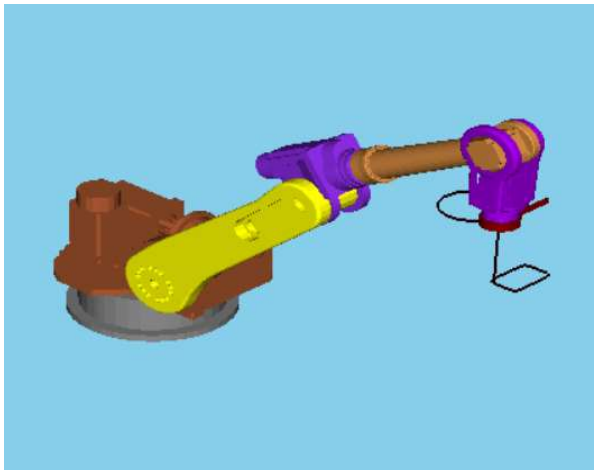
[Vx Vy Vz]

Hence we could solve for theta dot, which afterwards using equations of motion, we could solve for jerks.

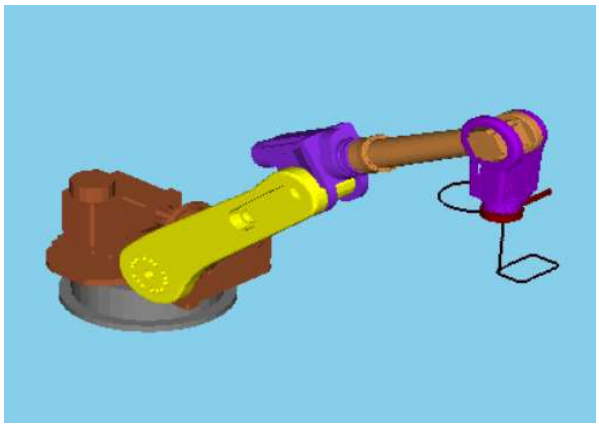
2.2.2.7 How we practically tackled the problem

Our final goal was to develop a user friendly software that can be used by any individual to design a path for the movement of the robot arm. Hence our choice gravitated towards using C# via Visual Studio for this purpose. First we tackled the mathematics explained above and moved onto graphics to test their accuracy. The robot arm was successfully simulated in our application. We used openGL to help with the graphics in the application.

First, we made sure that the path we are about to simulate was properly mapped in 3D space.

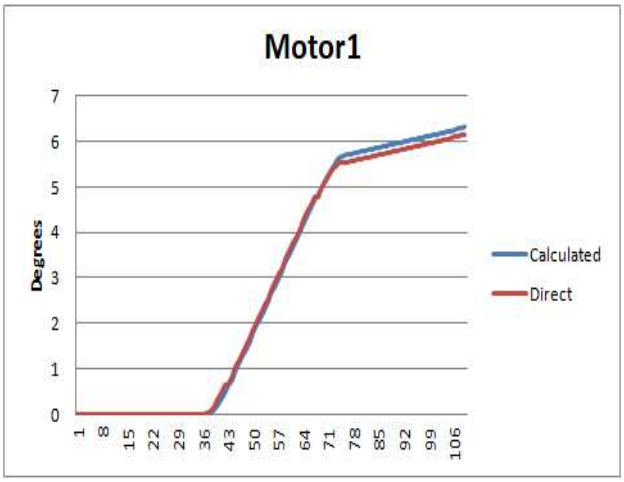


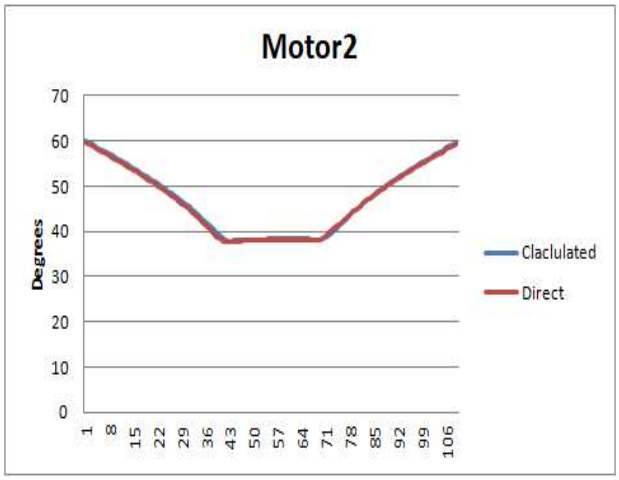
Then we calculated jerks, and reverse calculated the end effector positions using the jerks to verify our path.

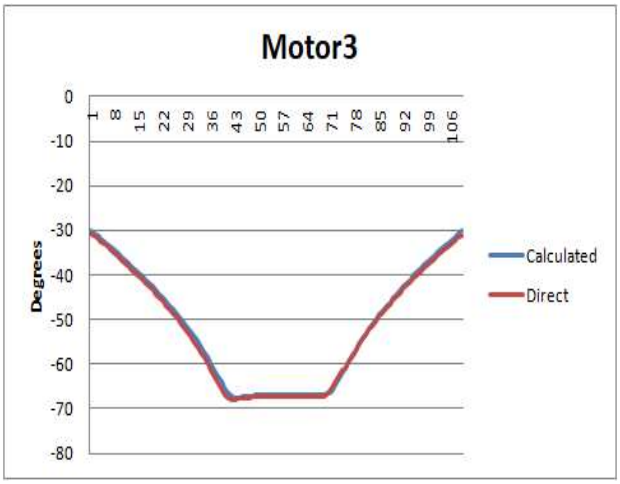


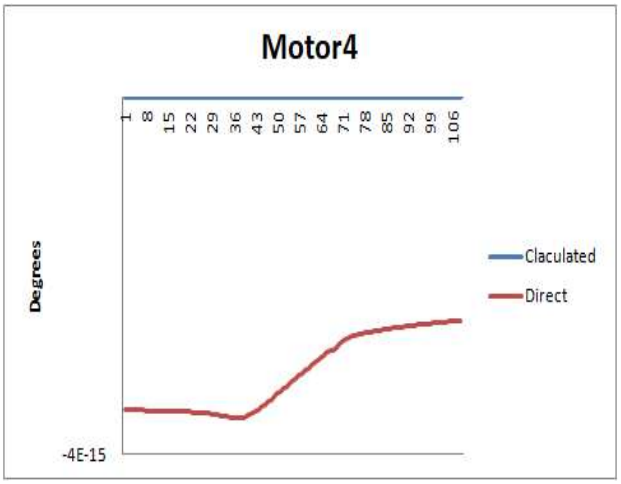
Following a series of fine tuning the constants used for calculations and data types used for storing variables, we were able to come to a state where the output was sufficiently similar to the expected output.

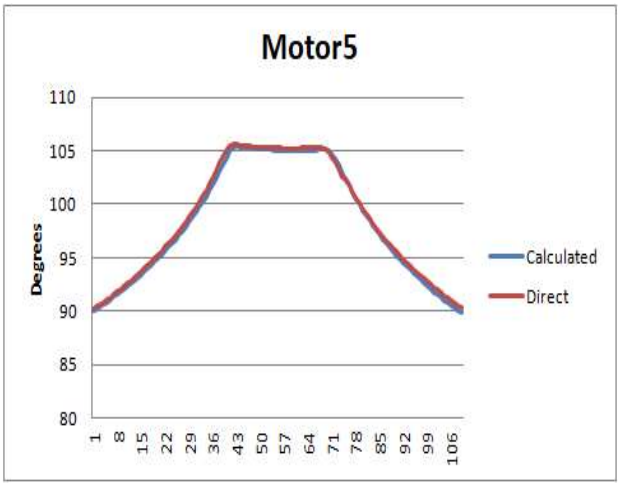
For a specific path, the comparisons of the angles in the motors are as follows.

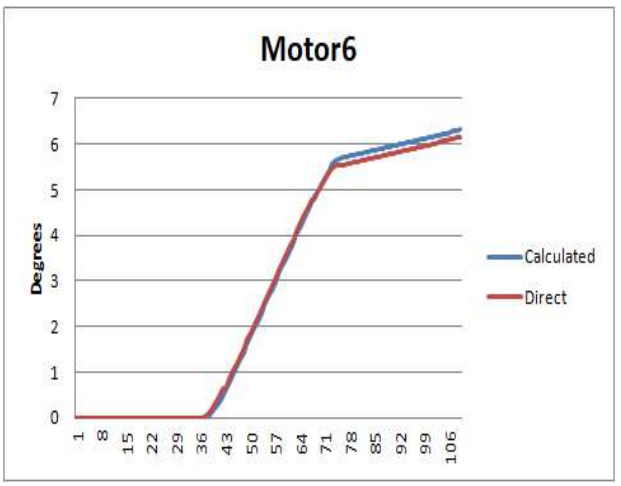




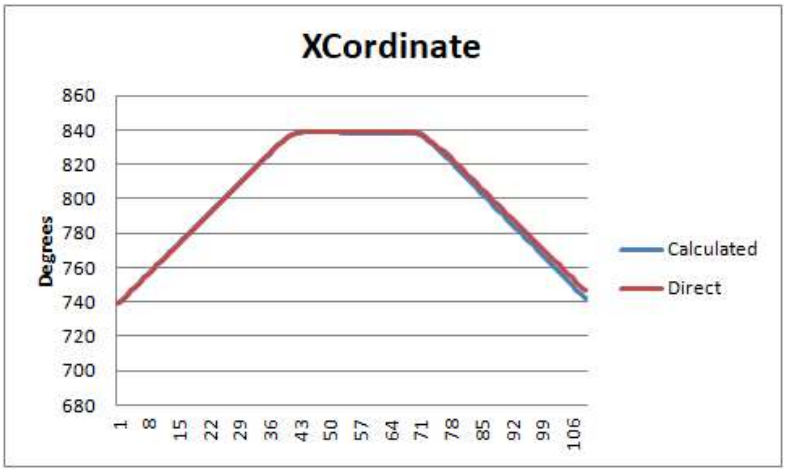


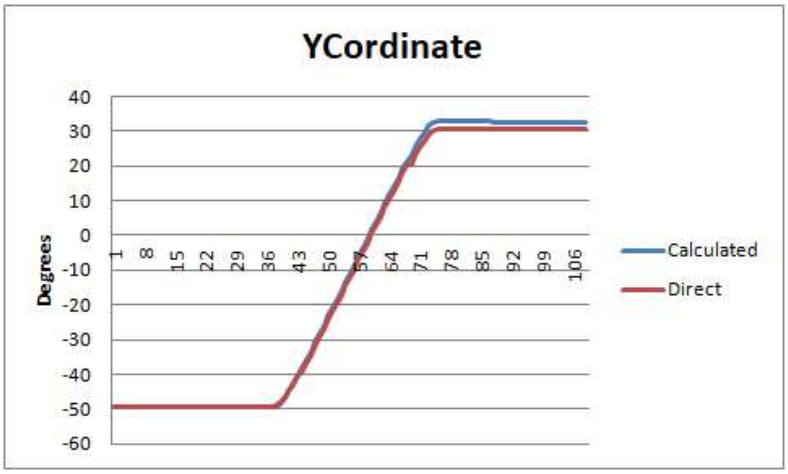


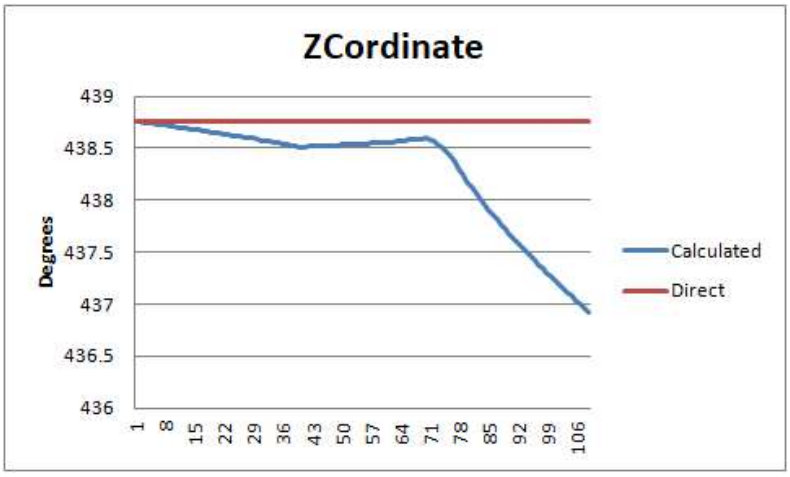




The deviation of the end effector coordinates are as follows,







2.2.2.8 G-Codes and M-Codes

Now that the main problem has been solved, our next task was to improve the practicality of the software. Our supervisor was keen on the robot arm being able to function according to G-Codes and M-Codes, as these codes are internationally recognised and will increase the usability of the robot arm.
G-Codes and M-Codes are geometric and machine related functions that are used to give commands to machines like the CNC. Some commonly used G-Codes are given below, out of them we used a handful, while also using a few undefined code numbers to invent our own custom code to suit the function at hand.

G00 - Positioning at rapid speed; Mill and Lathe
G01 - Linear interpolation (machining a straight line); Mill and Lathe
G02 - Circular interpolation clockwise (machining arcs); Mill and Lathe
G03 - Circular interpolation, counter clockwise; Mill and Lathe
G04 - Mill and Lathe, Dwell
G09 - Mill and Lathe, Exact stop
G10 - Setting offsets in the program; Mill and Lathe
G12 - Circular pocket milling, clockwise; Mill
G13 - Circular pocket milling, counterclockwise; Mill
G17 - X-Y plane for arc machining; Mill and Lathe with live tooling
G18 - Z-X plane for arc machining; Mill and Lathe with live tooling
G19 - Z-Y plane for arc machining; Mill and Lathe with live tooling

M00 - Program stop; Mill and Lathe
M01 - Optional program stop; Lathe and Mill
M02 - Program end; Lathe and Mill
M03 - Spindle on clockwise; Lathe and Mill
M04 - Spindle on counterclockwise; Lathe and Mill

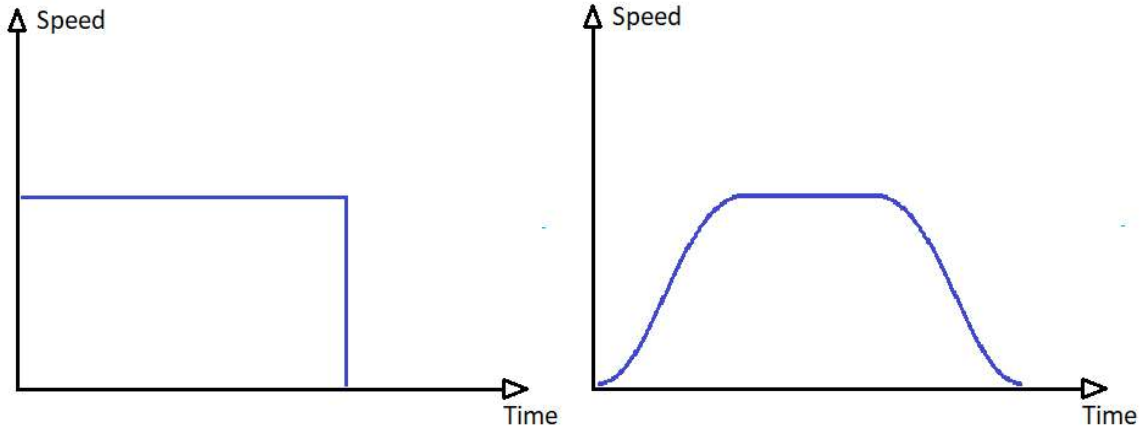
We included this part to the application such that user will be able to write a set of G-Codes using the software, which can be decoded by the software itself later on to create the required path. This option is very useful when the path needed to move in is externally generated. The G-Codes related to the path shown in the earlier figure is created by the application as follows,

```
G17
G02 X-10 Y10 Z0 I0 J10 K0 F20
G01 X-10 Y70 Z0 F20
G02 X0 Y80 Z0 I0 J70 K0 F20
G01 X30 Y80 Z0 F20
G02 X40 Y70 Z0 I30 J70 K0 F20
G01 X40 Y10 Z0 F20
G02 X30 Y0 Z0 I30 J10 K0 F20
G01 X0 Y0 Z0 F20
G18
G01 X0 Y0 Z150 F20
G17
G02 X0 Y0 Z150 I-50 J0 K150 F20
```

These codes will later be decoded to a set of geometrical shapes making up a path, which will later go through the aforementioned process to result in a list of jerks.

2.2.2.9 Smooth transitions to and from constant acceleration

In order to prevent slipping we had to actuate an acceleration phase from resting position to constant speed and another deceleration phase from constant speed to resting position. This will be shown by the end effector speed curve.



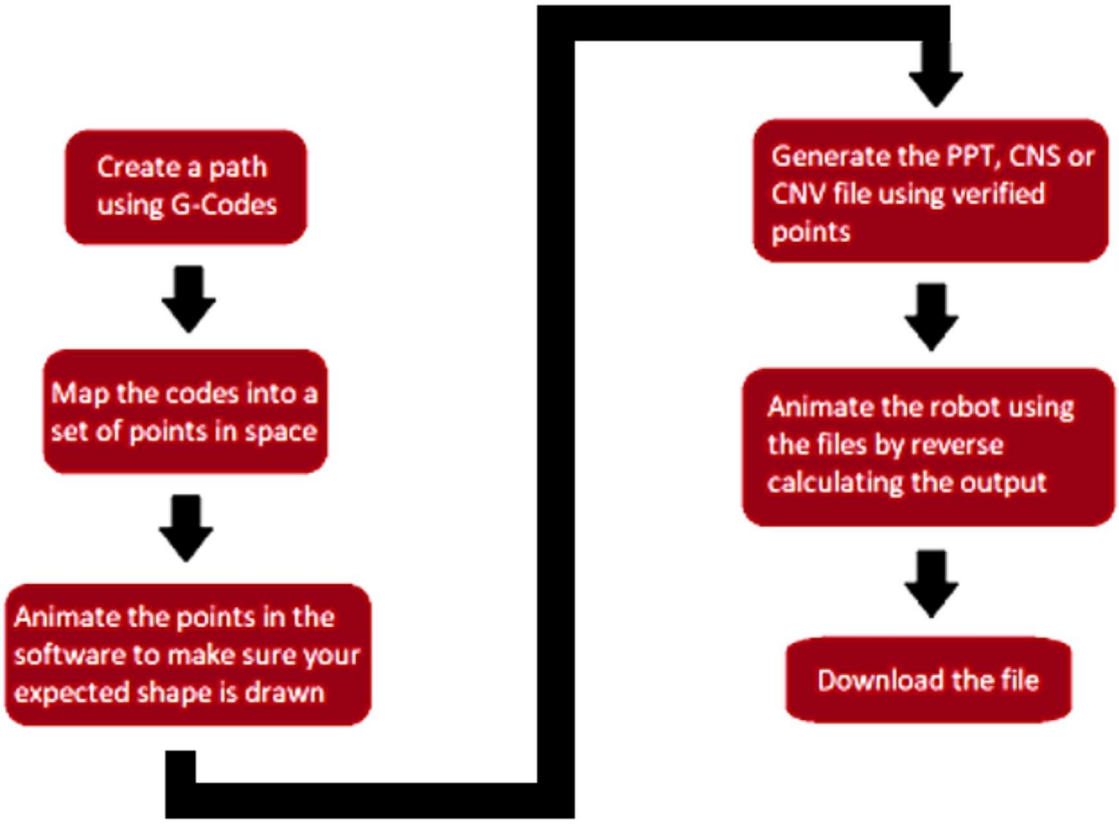
2.2.2.10 Testing with the actual robot

The actual robot has the capability of reading 3 file formats,

1. PPT - Point to point files will go through a course of start and stop via a number of points. This motion is not continuous.
2. CNS - Continuous files will follow the course of a set of points by bypassing the middle points. This motion is continuous but is not guaranteed to be constant speed.
3. CNV - Constant velocity motion files will follow a set path in a set velocity. It is the file format we adapted to our purpose.

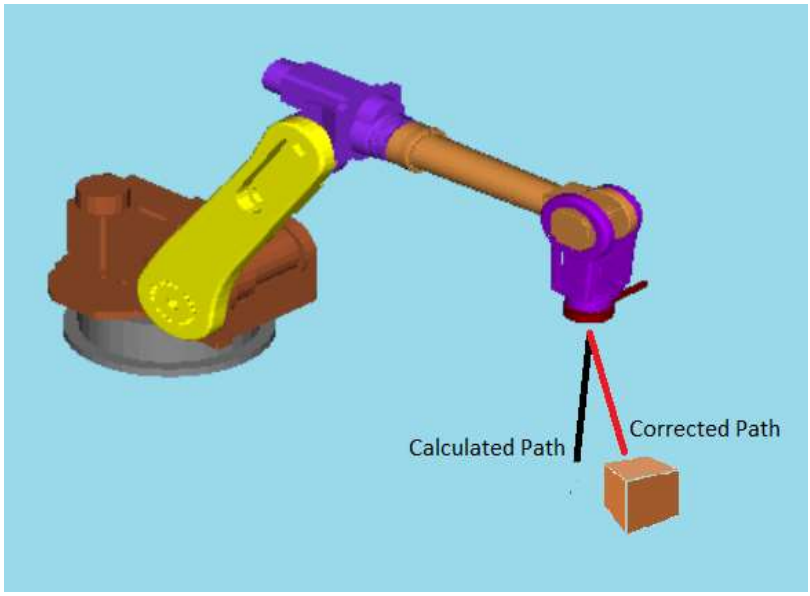
We were able to write all the above three file formats using our software and we implemented the said files in the robot. There were minute errors that were corrected.On the best case scenario we only have a 0.5 mm error for a 60 mm line.

To ensure that no harm comes to the robot by faulty lines, the software was designed such that a user can download a file onto the robot only if the following procedure is completed.



2.2.3 Protocol Interchanger

Now, the robot arm could move along a set path. But in case of an obstruction or a slight deviation, the arm cannot react accordingly due to lack of feedback. To overcome this issue, an ongoing project was focused on using a camera to detect and avoid obstacles and to calibrate the position in case of a deviation.



This relevant processing is done by a Raspberry PI. It was found that existing compatible USB cameras in unison with the Raspberry PI could not give a fast enough feed to give meaningful results. Also a USB cable could not extend over five feet according to USB standards, due to data corruption along the cable. Hence our first focus was to build a faster, more function specific and simple camera able to carry data for long distances.

2.2.3.1 Designing a Camera

The idea of a camera was simple. You have a sensor, a microcontroller, a lens and a handful of other components that work in unison to give an output. We identified two main types of sensors used in cameras. Their characteristics are as follows,

Description	CCD	CMOS
Camera components	Sensor+ Optic Support Chips+ Optics	Sensor+ Optic: Support Chips Sometimes
Speed	Moderate to fast	Fast
Sensitivity	High	Low
Noise	Low	Moderate
System complexity	High	Low
Sensor complexity	Low	High
Fill factor	High	Low
Chip output	Voltage (analog)	Bits (digital)
Pixel signal	Electron	Voltage
Uniform shuttering	High to moderate	Low

Our task required a sensor with high speed and low power. The quality of the picture was of less importance to us. Hence we figured the most suitable sensor would be the CMOS sensor.

Now that a sensor was selected we moved on to the microcontroller. We decided to use an atmel microcontroller. It is with the lens that we found a couple of issues. For one, choosing a lens for a sensor is very difficult and a custom lens is very expensive. Also as our task required focusing on extreme close ups and also moderate distances, we might need to adjust the lens curvature, which could be achieved by a couple of lenses and a complex mechanism. This required precise placement, leading to the fact that a very small but accurate motor might be required in the process. Ultimately we came to the conclusion that the cost incurred is not worth the benefits of a custom camera. Hence we moved on to the second option.

2.2.3.2 Choosing a Wifi Camera

Our second solution was to simply use a WiFi camera. This will solve the issue of cabling, but not the issue of feed rate. Also if a wifi camera was used we had to synthesize the data before sending it to the Raspberry PI. So, using an expensive WiFi camera to achieve the results which can be cheaply achieved by using a USB camera was not the best choice, especially when considering the next solution.

2.2.3.3 Designing a Protocol interchanger

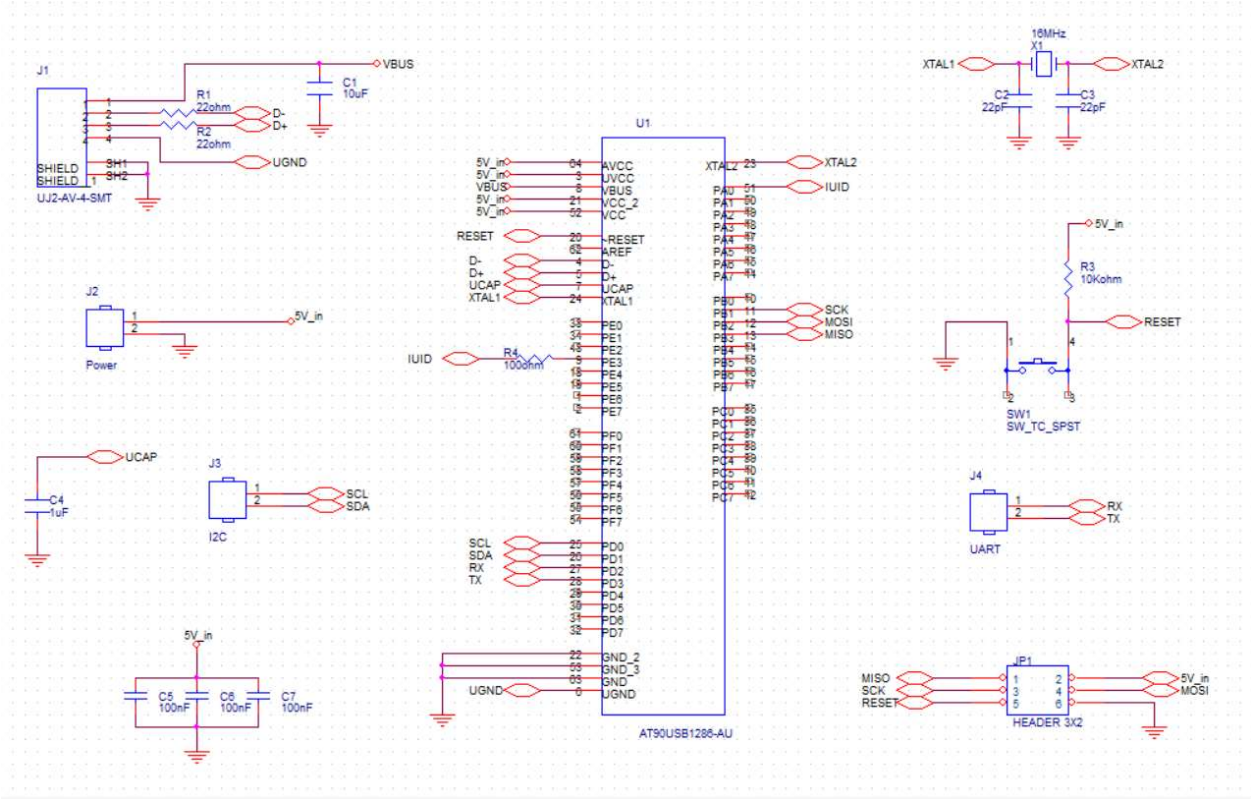
The third solution we came up with was to build a protocol interchanger. We will be using a standard USB camera, which will be fed into the interchanger. Then it will change the protocol in the data such that it is customized to our specific needs. It must be understood that the USB protocol is huge. Our task required grabbing photographs, which, as known, is not crucial as numerical data. A glitch in a photograph is almost unrecognizable.
Hence, most of the protocol used to transfer data via USB is redundant to our purpose, given that it is heavily inclined towards error free transmission.
Our solution is as follows,
We feed the output from the USB camera to the interchanger. Then it is disassembled, and rearranged to a custom protocol we built. This in turn is sent to the Raspberry PI.

This solution in no way addresses our initial problem of a low feed rate. But ultimately our goal is to have fast enough results to avoid unexpected collisions.
So instead of observing from the camera point of view, we decided to tackle the problem from the Raspberry PI point of view. The OS of the Raspberry PI definitely slows us down due to its various layers. While an OS is useful for a general purpose board, for our function specific task, it is a bit redundant. Hence we decided to write the processing of the photograph feed via bare metal. In order to de so, if we were to use a USB camera, we had to write the USB protocol using bare metal too. Which, it turns out is a huge task. After a few days of trying to understand the protocol thoroughly, we decided that it is best to have an intermediate with built in USB. Hence spurned the idea of the protocol interchanger.
We used a AT90USB1287 for this task. It was capable of acting as a USB host and a slave, as it is clear from the above figure that it needs to act as both.

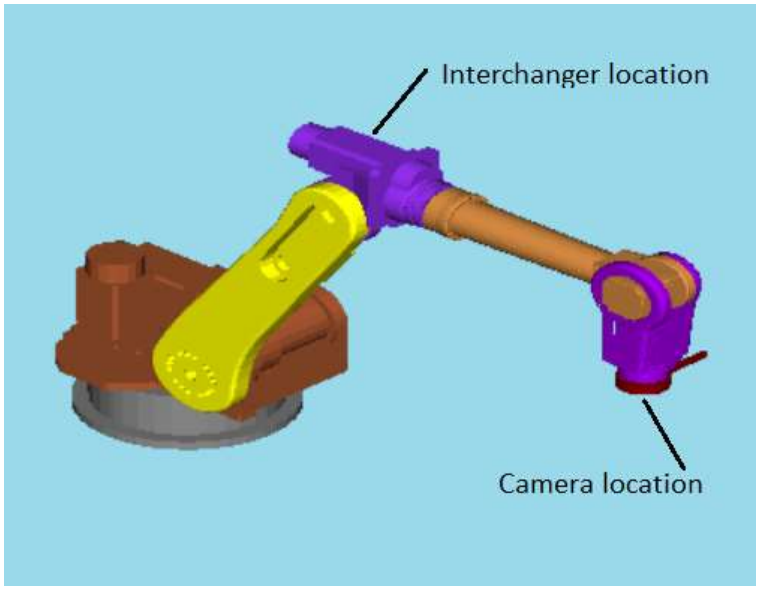
The next choice we had was to decide which protocol is best suited to communicate with the Raspberry PI: SPI, I2C or UART. The characteristics of these protocols are as follows.

Universal asynchronous receiver-transmitter (UART)	Serial Peripheral Interface (SPI)	Inter-Integrated Circuit (I^2C)
asynchronous, no clock needed	synchronous, needs clock	synchronous, needs clock
Simplex, Full duplex, or Half duplex	Full duplex	Half duplex
single-master single-slave, limited to one sensor	single-master multiple-slave	multiple-master multiple-slave, theoretically limited to 127 sensors
Baud rate (bps) 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200	implementations go over 10Mb/s	limited to 1Mb/s (fast mode) and 3.4Mb/s (high speed mode)
2 data lines (RX, TX)	4 data lines (SLCK, MOSI, MISO, SS)	2 data lines (SDA and SCL)
no need for a Chip Select (SC) line	needs a Chip Select (SC) line	no need for a Chip Select (SC) line
RX and TX data lines are bidirectional	MISO and MOSI data lines are bidirectional	SDA and SCL data lines are bidirectional
Slowest but simple to implement	Fastest but complex when integrating more than one sensor (master has to reconfigure itself each time it needs to communicate with a different slave)	Fast and simple to implement. The physical bus implementation enables the master devices to simultaneously write and listen to the bus lines.

In order to keep the flexibility we decided to allow all three outlets from the interchanger. The finished schematic looked similar to this,



The second issue will not be solved by this method. So we decided to fix the interchanger on the robot arm itself, so that the USB cable length is safely below 5 feet.



2.2.4 Human Machine Interface(HMI)

The current HMI used on the robot arm is a costly one.

While it is suitable for a test case, given the fact that the robot arm is marketed towards small and medium scale factories, it is a point where the cost can be reduced. A complicated HMI is not a requirement for the robot as most functions are to be done via the PC and only the implementation of paths is left to the HMI.
Hence the third project we got was to design a cost effective HMI. We were able to build one for about Rs.6000, which was a great achievement

2.2.4.1 How does an HMI work

An HMI acts as an easy interface between a machine and a user. It has a human understandable front, and another front which is machine understandable. An HMI pretty much acts as an intermediate between humans and machines, hence the name, Human Machine Interface.

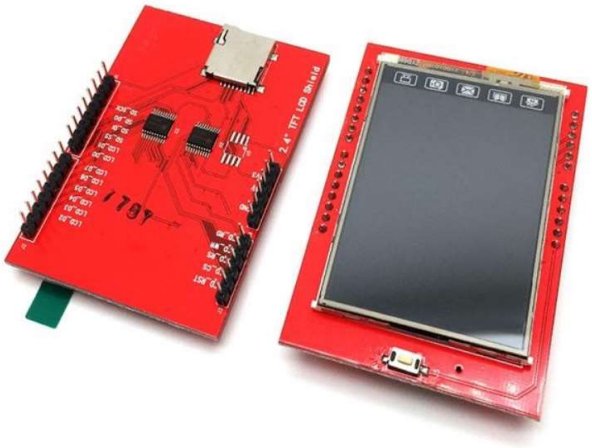
2.2.4.2 Components in our HMI

A traditional HMI has the following components

1. A screen
2. An input source (keypad or touch screen)
3. A microcontroller

For our HMI we decided to use the following components:

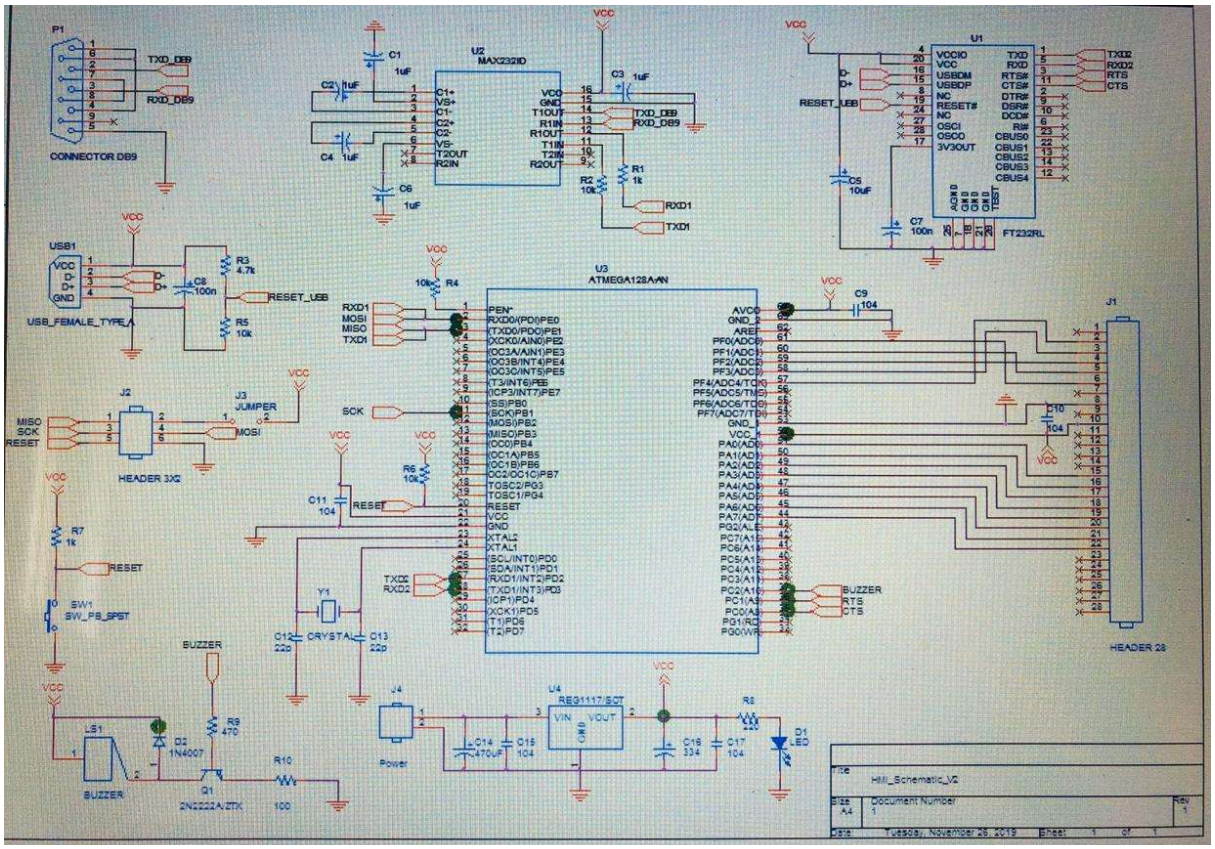
If we had decided to use a screen and a keypad separately, the flexibility of the input options will be reduced. Also the price of a normal screen vs a touch screen is alarmingly similar. So we decided to use a touchscreen for our input and display purposes in order to increase flexibility and reduce cost. The touchscreen we used was 2.8 inch TFT LCD with touchscreen.



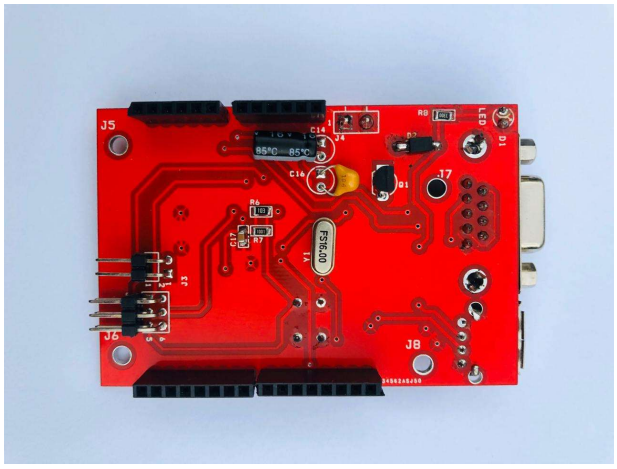
For the microcontroller we chose an Atmel microcontroller, atmega 128. The microcontroller had all the basics that we needed and more, and it is easy to integrate with the robot as the robot too used Atmel microcontrollers.

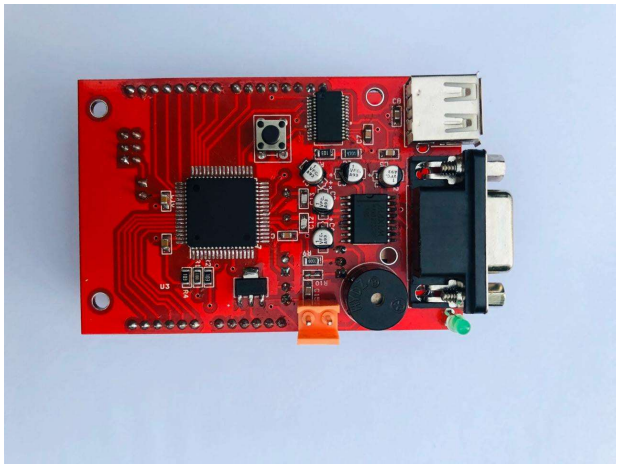
2.2.4.3 Designing the circuit

First we designed a circuit for the HMI. The circuit was designed to act as a shield for the touchscreen in order to reduce the connections and for compact design.



The finished circuit looked like this,





2.2.4.4 Designing the software

Following this we had to design the software that is to be run on the microcontroller. We used atmel studio for the coding and eXtreme Burner for uploading the codes to the microcontroller. We took the help of a few libraries available on touch screens and modified them according to our needs. The main library we used was LCDWIKI_GUI and LCDWIKI_KBV. They were written to suit the atmega32p microcontroller, but we adjusted it accordingly.
We went to a length to prevent the use of interrupts in the program as many forums advised against the practice in the case of a touchscreen. Our basis was that the readings of the touch will be taken every few milliseconds and accordingly the next step will be directed to a case condition within the main loop, instead of exiting the main loop as in the case of interrupts. We faced a number of issues in the process. One such issue being that the screen coordinates were not properly calibrated. Hence we had to manually calibrate the screen. The next issue was the frequently misleading readings occuring once every few minutes. To overcome this, we found the most occuring reading over a period of time to find the exact reading. This meant the response time was reduced, but it was not all too obvious. The third issue we faced is the refresh time of the screen. The time taken to repaint the screen is very prominent to the human eye, and is not aesthetically pleasing at all. As this is an issue with the hardware we selected, we could not come up with any suitable solution for this. But as this is a low cost project we overruled the cost of better hardware over the drawback of the refresh time.

2.2.4.5 Designing the enclosure

After the software was tested and is running smoothly to the human touch, we proceeded to finish our product with a suitable enclosure. We decided that a simple enclosure is suitable much similar to the HMI we already had. Caution was taken to make sure that the enclosure was moldable, so that in case the mass production of the product is initiated, we could use injection molding for the purpose.