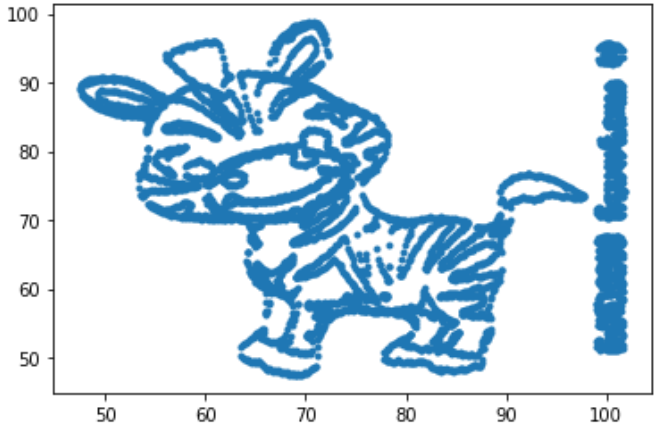
Simulation of Multi Axis Robotic Arm



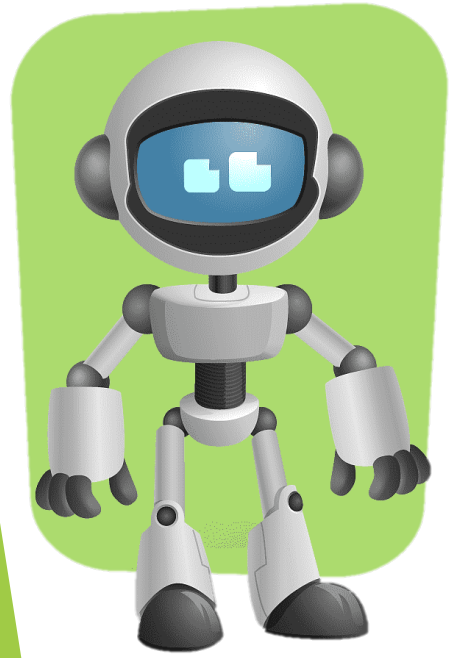
Agenda

- Introduction
- Abstract
- Literature Survey
- Existing Model
- Major challenges faced
- Proposed Solution
- Software Requirements
- Coding Algorithm
- Advantages and Disadvantages
- Applications
- Results
- Conclusion

Introduction

"We are fascinated with robots because they are a reflection of ourselves."

–Ken Goldberg



- The era of 21st Century is a new age for robotics.
- In our imagination a robot is a machine that looks and acts like a human being.
- Robots are defined as, man-made mechanical devices that can move by themselves, whose motion must be modelled, planned, sensed, actuated and controlled, and whose motion behavior can be influenced by programming.
- Types of Robots:
  - Industrial Robots
  - Domestic and household robots
  - Military robots
  - Space robots etc…

Abstract

- Robotic arms plays a vital role in the industrial and automobile sector.
- The current working of robotic arm in plotting sector is based on time, velocities and accelerations of the motors.
- While working on velocities and time, precision and accuracy plays a key role. So, cost factor and code complexity increases.
- Proposed system converts G-code into number of steps and motor control parameters of the stepper motors, in order to reduce the code complexity.
- Simulation of the proposed algorithm is obtained using Python 3.

Literature Survey

Ref	Authors	Title	Source	Findings
[1]	J. Liu Q. Luo	Modeling and Simulation of Robotic Arm in MATLAB for Industrial Applications	IEEE (2019)	Simulation of robotic arm in MATLAB based on motor feed
[2]	Z. Wei Y. Li	Design and Simulation of Robotic Arm with Light Weight and Heavy Load	IEEE (2019)	Multi axis approach and arm parameter designing
[3]	M. M. Hasan M. R. Khan A. T. Noman H. Rashid N. Ahmed	Design and Implementation of a Microcontroller Based Low Cost Computer Numerical Control (CNC) Plotter using Motor Driver Controller	IEEE (2019)	CNC workspace design and generating motor driver code
[4]	Groover M P	Industrial Robotics	Pearson Edu publication	Robotics basics

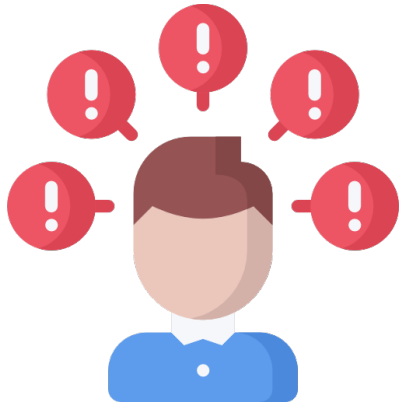
Existing Model

MATLAB®

- Read the G-Code file into memory
- Interpret the G-Code
- Plan the continuous path
- Add time dimension (velocities)
- Generate converted motor code
- Motor drivers

Major challenges faced

- MATLAB (Licensed software)
- Difficult to add time dimension
- Code complexity
- High precision and high accuracy motors required
- Cost of robotic arm increases



Proposed Model

python

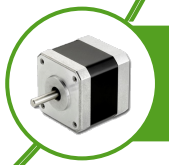
Read the G-Code file into memory

Interpret the G-Code

Plan the continuous path

Calculate steps (Based on distances)

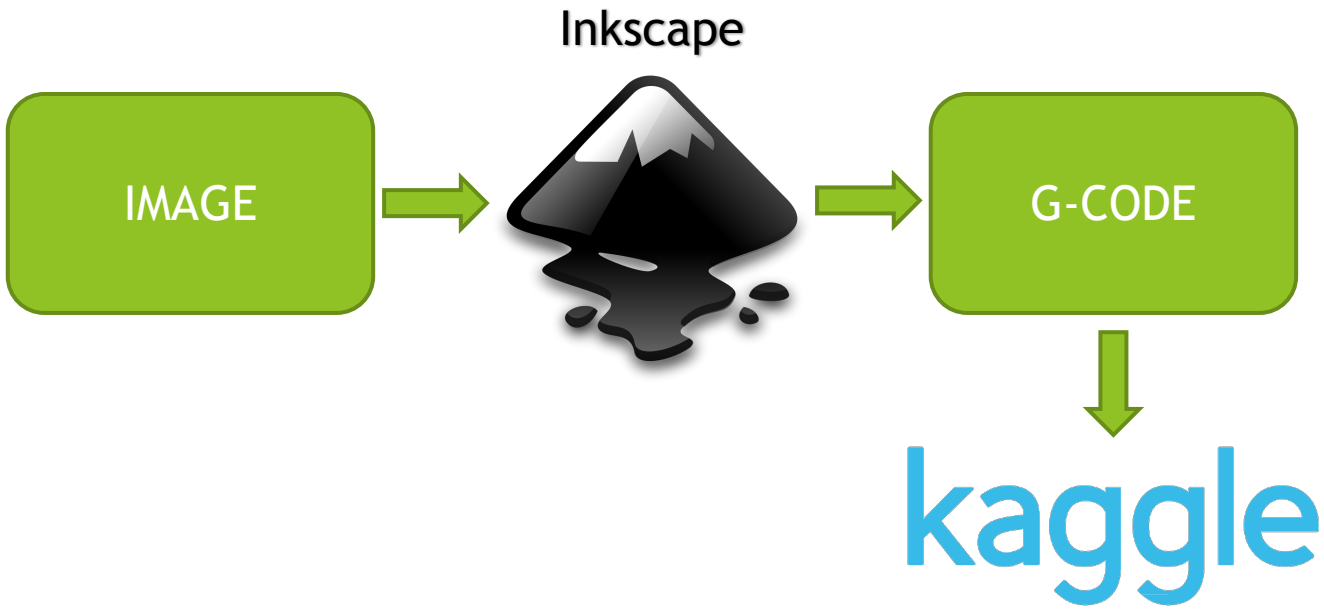
Generate converted motor code



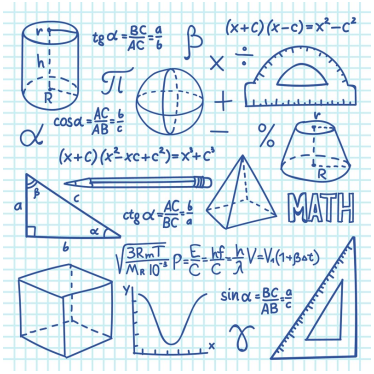
Motor drivers

Software Requirements

- Inkscape (Preloaded with Gcode tools)
- Python 3 (Kaggle)



G-code



- G-code stands for Geometric code.
- It is a programming language for Robotic Arms.
- G- Code generation takes images as input, finds path by obtaining co-relation between every two points and establishes a link, specifying the same relation in form of geometry coordinates.
- There are many ways to generate G-code from images. Such as online generators or G-Code generation software's.
- Inkscape is one such software which is used to generate G-Code.

```
G01 X64.685354 Y91.178443 Z-1.000000
G03 X63.341876 Y91.107433 Z-1.000000 I-0.400783 J-5.161893
G03 X63.273962 Y90.917012 Z-1.000000 I0.021278 J-0.114910
G02 X63.496924 Y90.471635 Z-1.000000 I-0.804890 J-0.681437
```

Cmd	Meaning
G00	Rapid Positioning
G01	Linear Interpolation
G02	Circular Interpolation Clockwise
G03	Circular Interpolation Anticlockwise

G-Code Commands

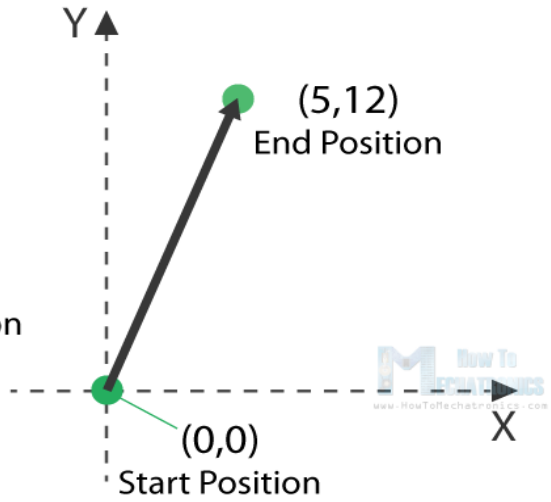
G00 - Rapid Positioning

Example:

G00 X5 Y12

Rapid motion

Set end position



G01 - Linear Interpolation

Example:

G01 X5 Y12 F200

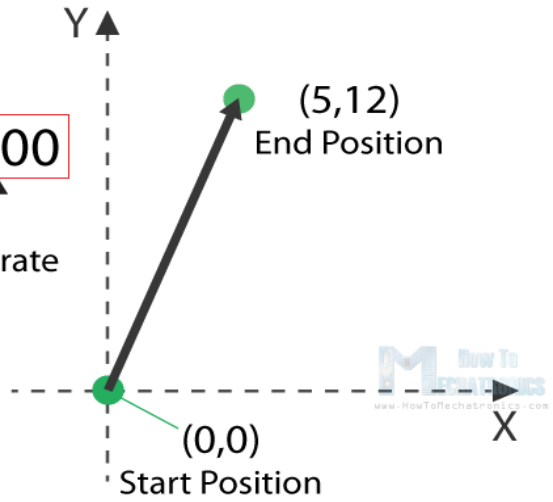
Linear Motion

Set end position

Feed rate

If units = G20 - Feed rate is in/min

If units = G21 - Feed rate is mm/min



G02 - Clockwise Circular Interpolation

Example:

- G01 X5 Y12 F200
- G02 X10 Y7 I0 J-5

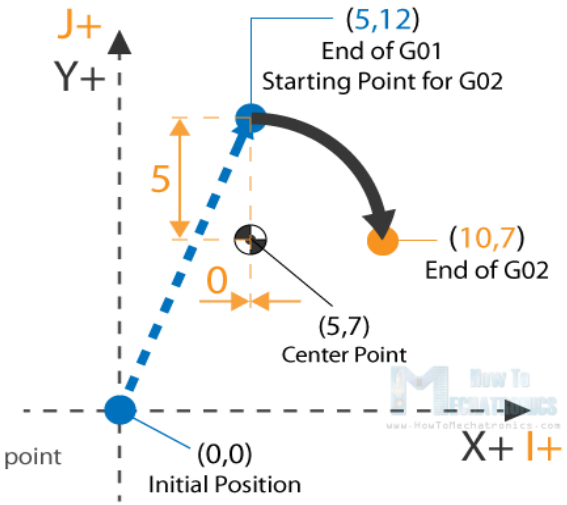
Circular motion

Set end position

X offset to center point

Y offset to center point

X and Y offests are relative to starting point



G03 - Counterclockwise Circular Interpolation

Example:

- G01 X5 Y12 F200
- G03 X0 Y7 I0 J-5

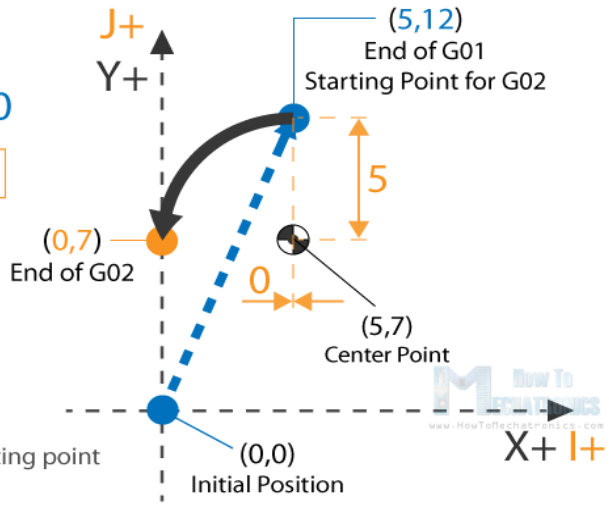
Circular motion

Set end position

X offset to center point

Y offset to center point

X and Y offests are relative to starting point



Why Kaggle?

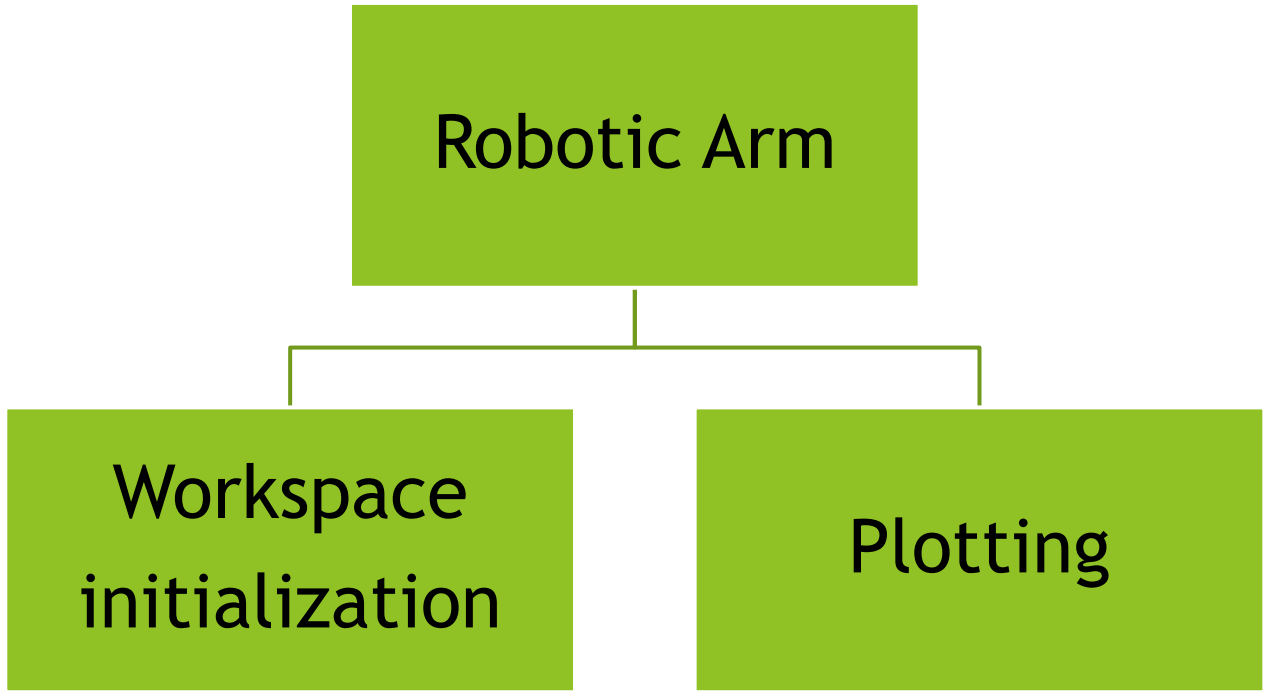
Basic PC

- Dual core
- 4gb RAM
- No SSD
- No GPU
- No TPU
- Local environment

kaggle

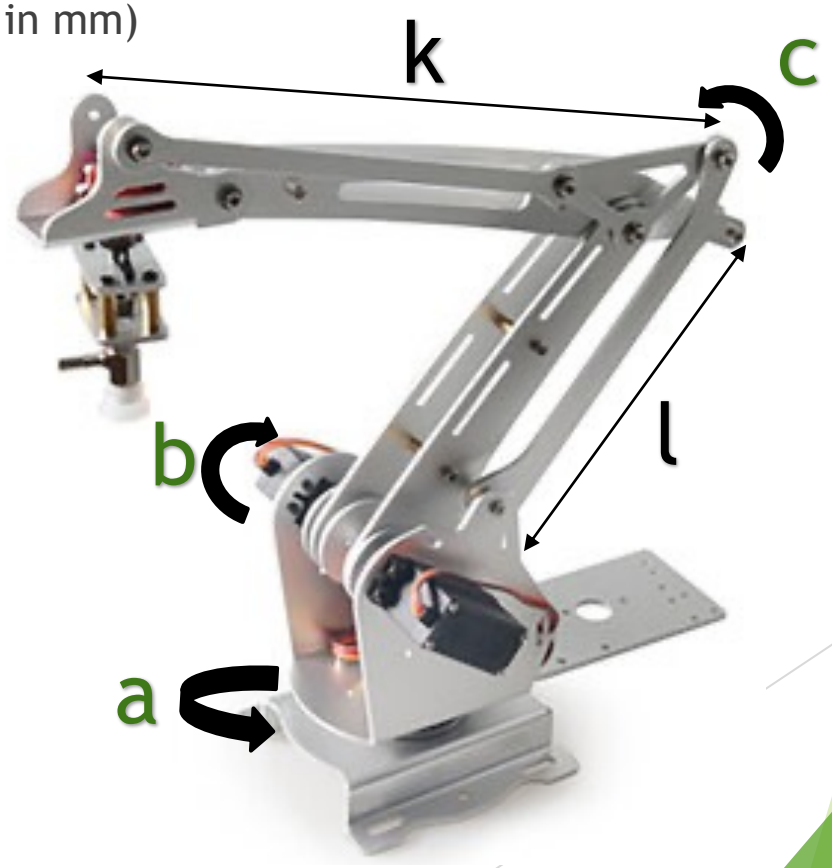
- Quad core
- 16gb RAM
- 20gb auto save (per user)
- Nvidia Tesla P100 GPU
- TPU v 3-8
- Virtual remote environment

Coding Algorithm



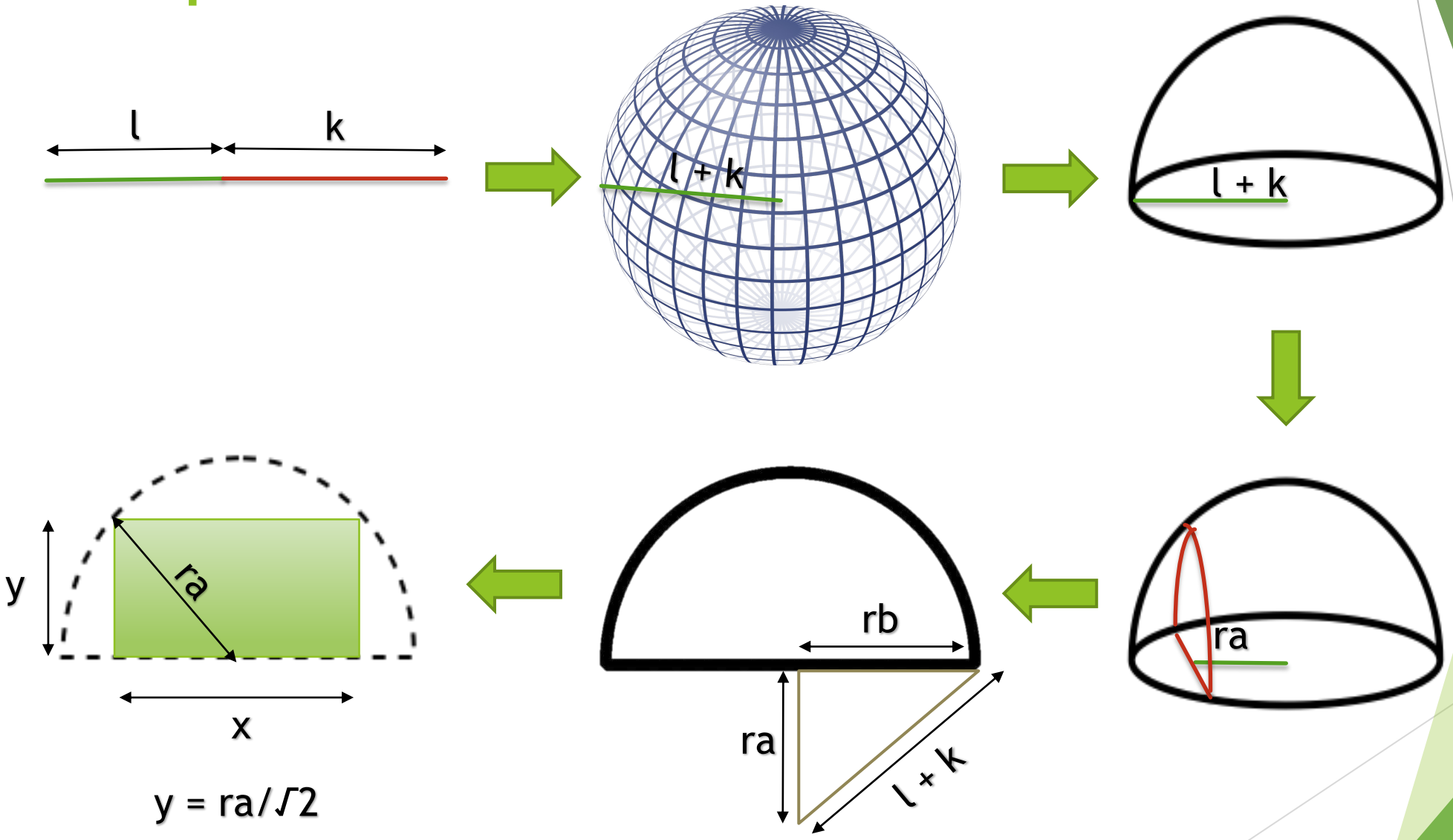
Workspace Initialization

- Develops a 2D workspace based on robotic arm physical parameters.
- INPUTS:
  - Arm lengths (l and k in mm)
  - Distance between base and workspace (ra in mm)
  - Stepper motors step size (in degrees)
- OUTPUTS:
  - Workspace coordinates file
  - Workspace dimensions



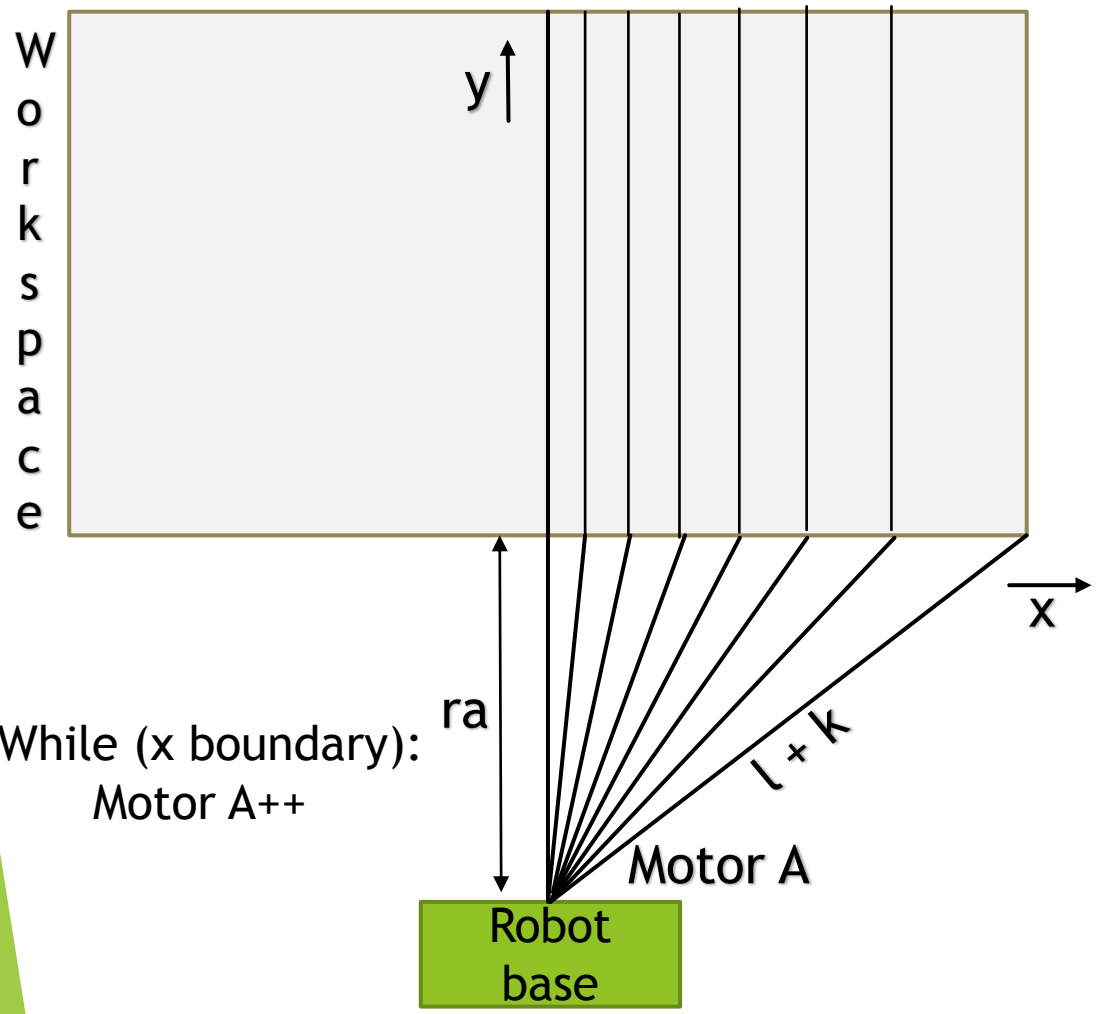
Note: ra<l+k

Workspace Initialization



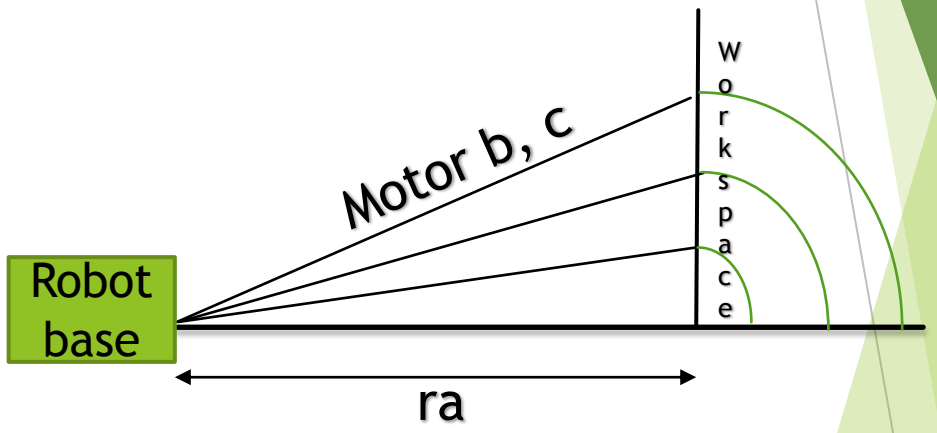
Workspace Initialization

For X Coordinates



While (x boundary):
Motor A++

For Y Coordinates



While (y boundary):
Motor C++
if (arm crosses workspace):
while (arm not on workspace):
Motor B++

Workspace Initialization

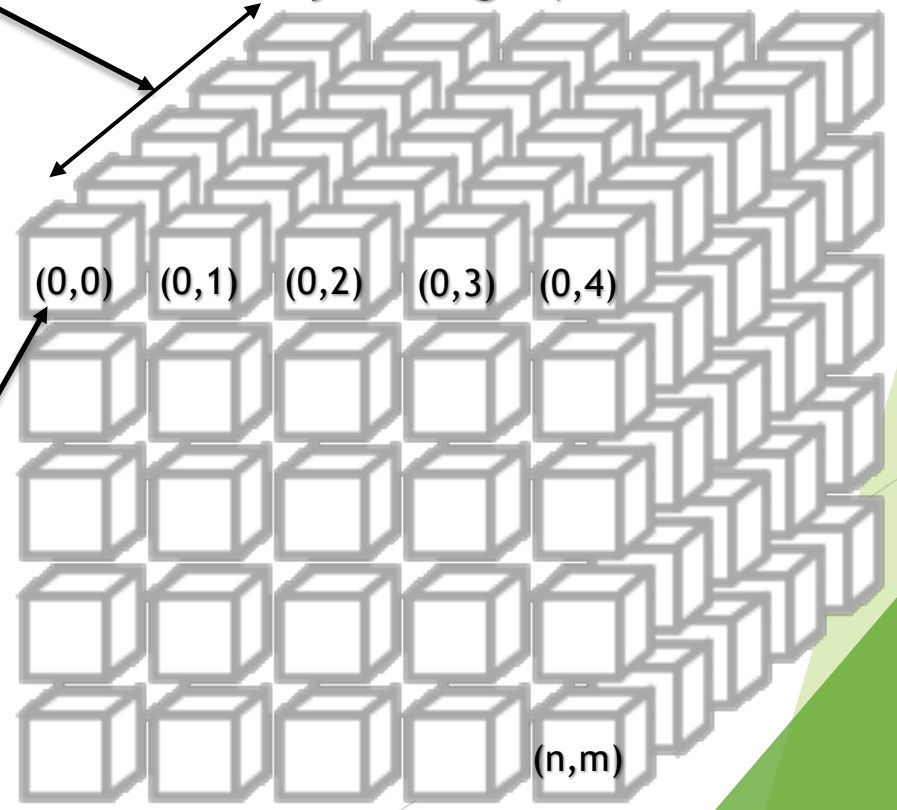
Values stored at every coordinate where y=0

No. of steps made by Motor B	No. of steps made by Motor C	Y Displacement	X Displacement	No. of steps made by Motor A	No. of Y's in that X
0	1	2	3	4	5

Values stored at remaining coordinates

No. of steps made by Motor B	No. of steps made by Motor C	Y Displacement
0	1	2

Memory storage (Data Structure)

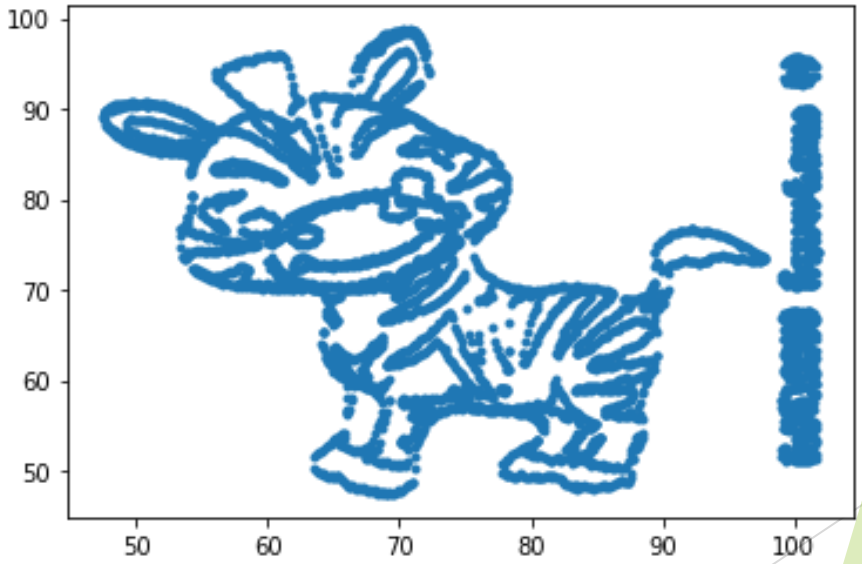


Tag

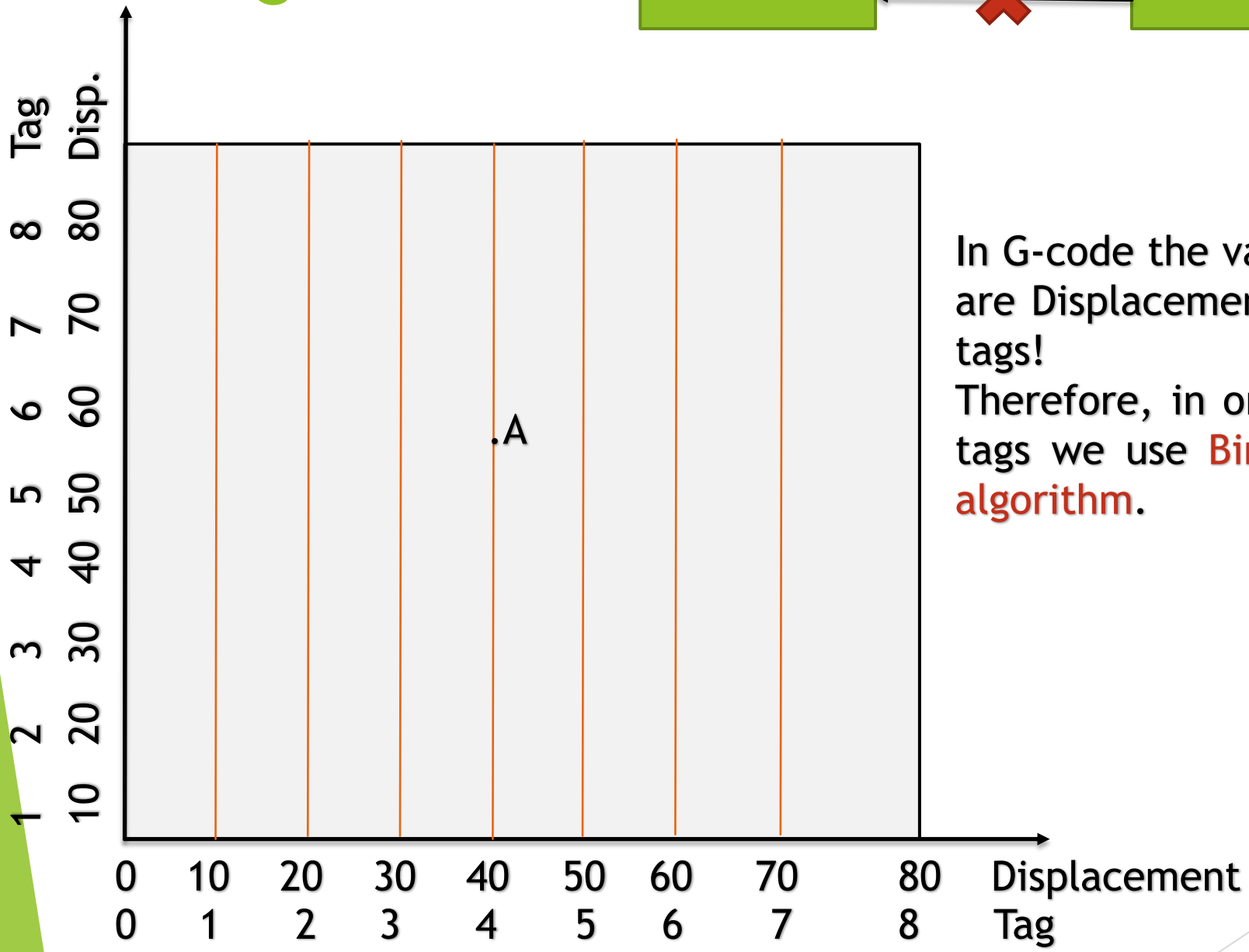
Accessing Tags
Ex: [5,6,3]

Plotting

- Generates simulated 2D Plot and motor driving code, using G- Code and workspace.
- INPUTS:
  - G- Code (which is generated using Inkscape)
  - Workspace initialization code outputs
- OUTPUTS:
  - 2D Plot of image
  - Motor Driving code
  - Coordinates list (points which arm plotted



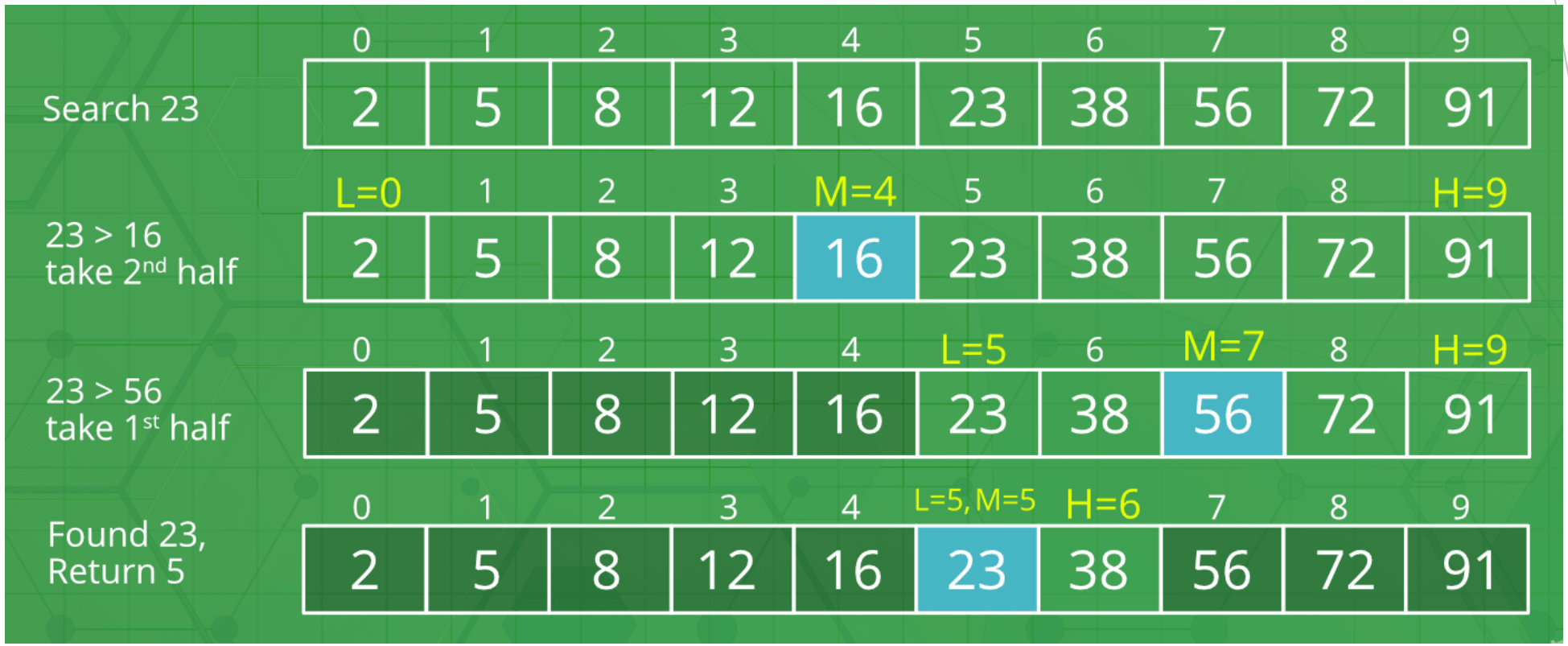
Plotting



In G-code the values stated are Displacements. But not tags!

Therefore, in order to find tags we use Binary Search algorithm.

Binary Search Algorithm



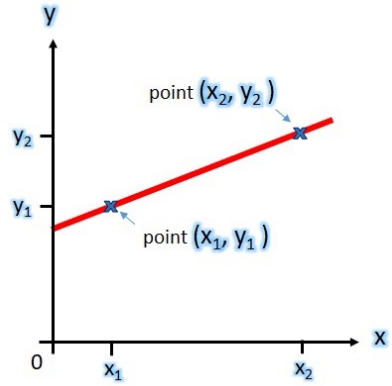
In our context, Binary search is performed based on tags and at every point displacement value is cross checked with the displacement value of G-code.

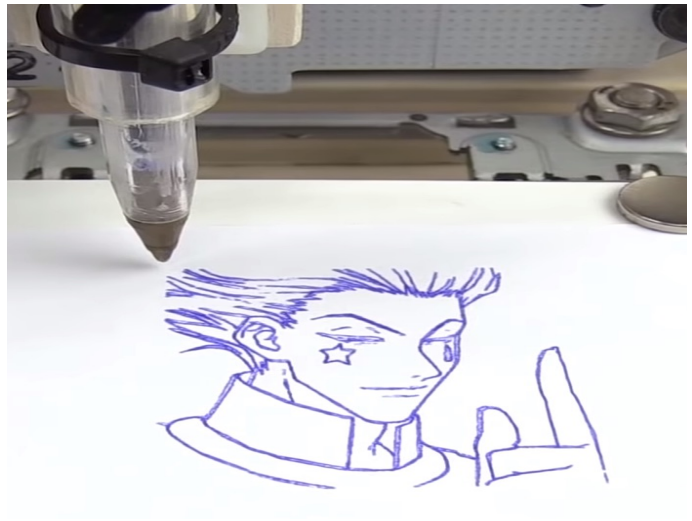
This process is repeated until the closest approximation of displacement tag is obtained.

Plotting

Line Equation (G01)

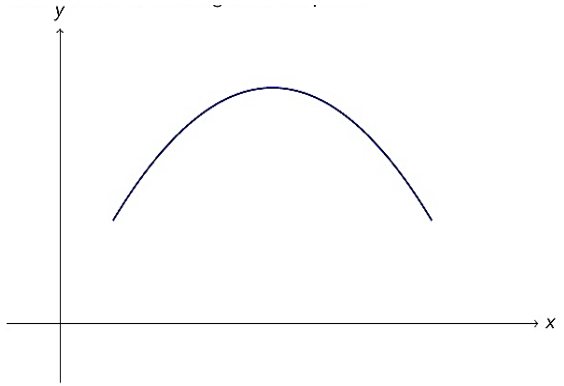
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$





Curve Equation (G02 and G03)

$$(x - h)^2 + (y - k)^2 = r^2$$



h = x1 + i

k = y1 + j

i and j values are taken from G-code

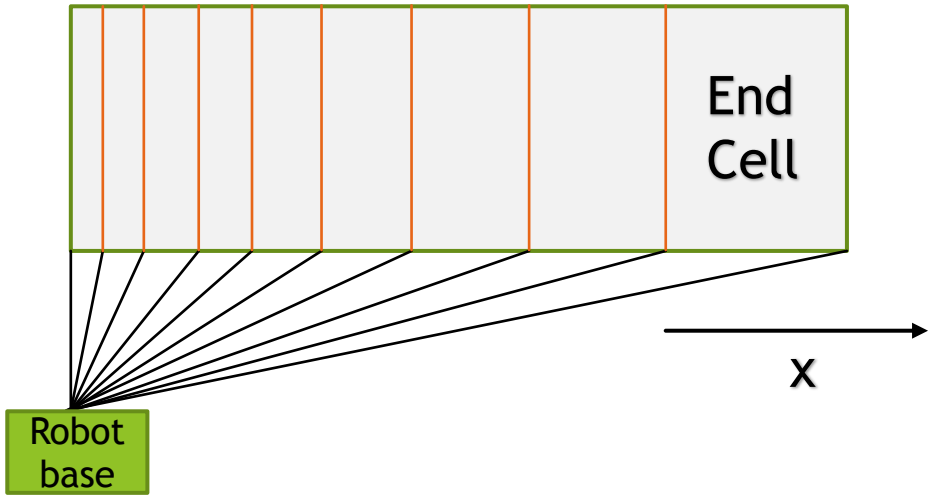
Binary Search is performed on every consecutive points and are joined.

Advantages

- Comparatively reduces robotic arm cost.
- Python and inkscape are open sources.
- Saves a lot of time.
- Reduced code complexity.
- User friendly.
- Robotic arm parameters and workspace can be altered.
- Doesn't need time factor as input.

Disadvantages

- Comparatively low accuracy and precision.
- Cell size increases as graph approaches closer to boundary limits.
- G-Code must be of desired format (Gxx) else errors raise.
- Chances of job loss in various sectors.



G-code

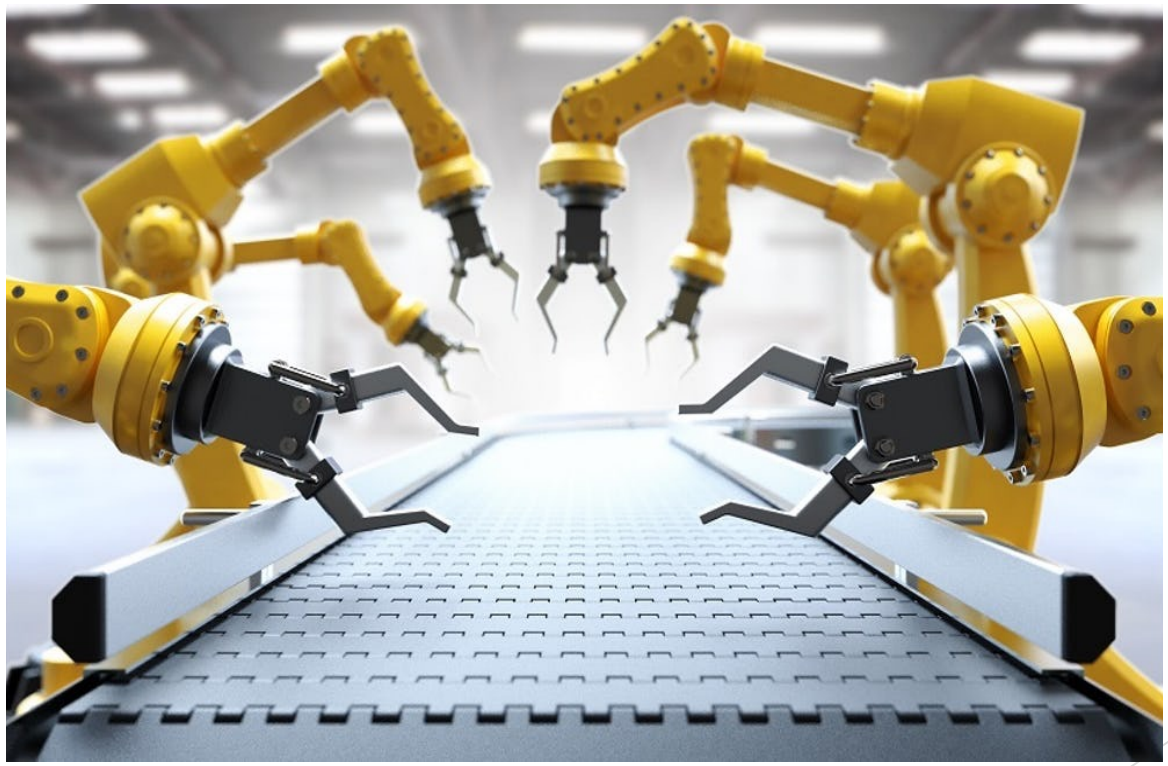
G01

G1

Applications

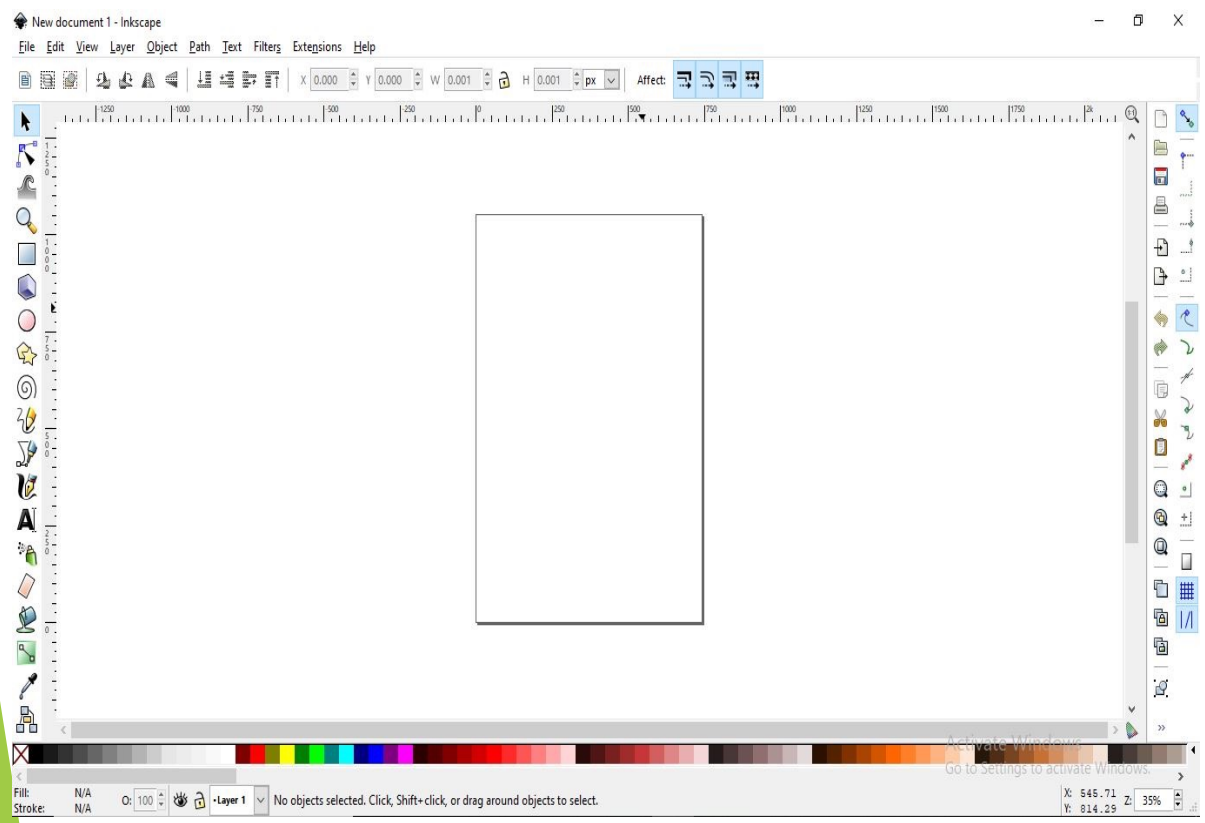
The following are various applications of Multi Axis Robotic Arm:

- 2D Sketching
- Arc welding
- Spot welding
- Painting
- Metal cutting
- Industrial needs
- Automobile manufacturing
- Home needs

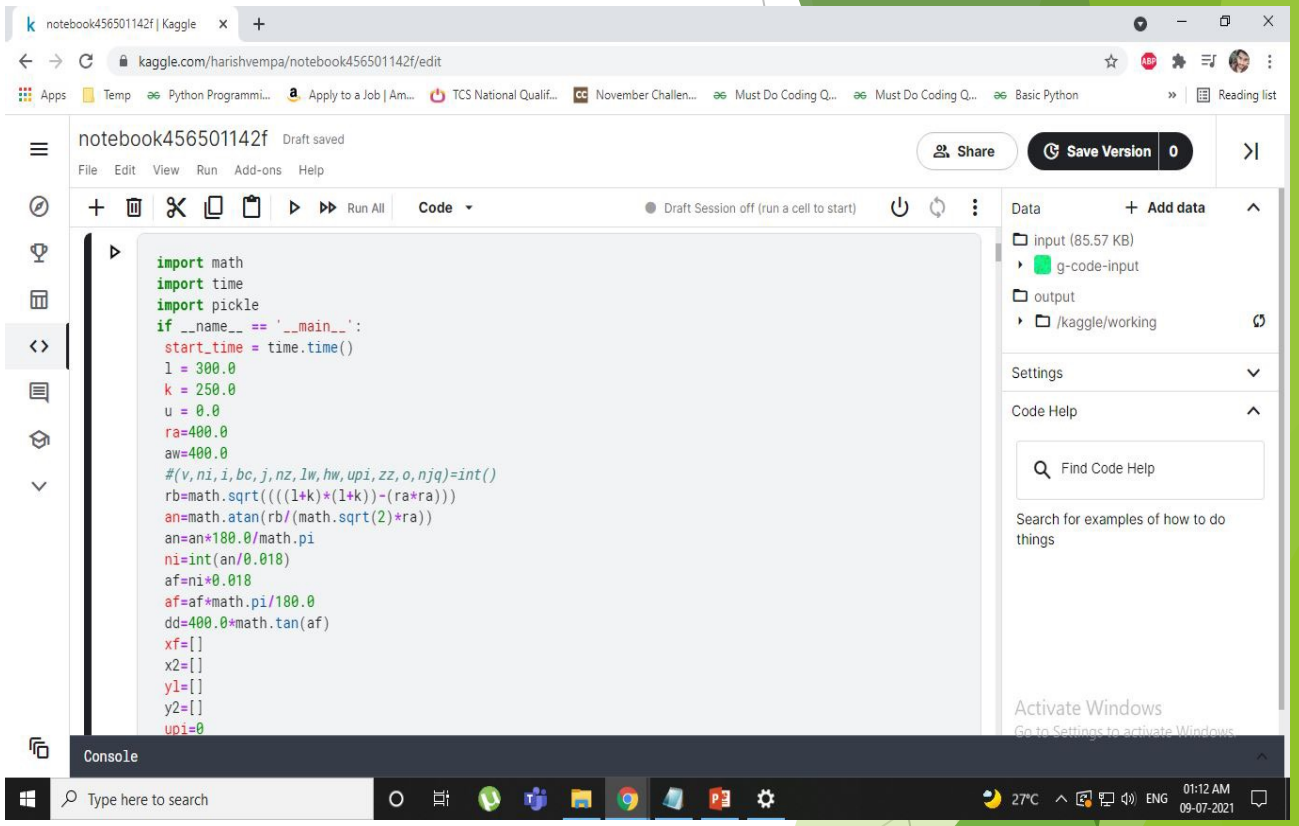


Working Environments

Inkscape



Kaggle



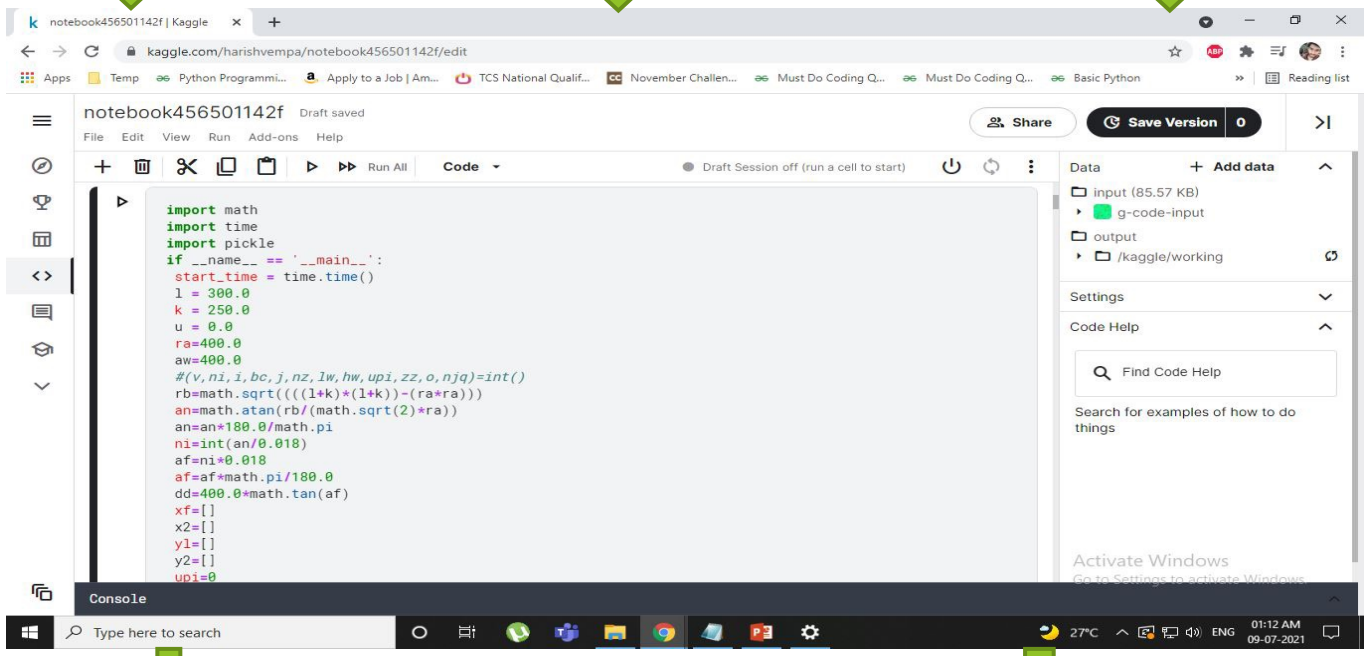
kaggle

Results (Code-1)

Inputs

```
0.018
ra=400.0
l = 300.0
k = 250.0
```

Workspace Initialization



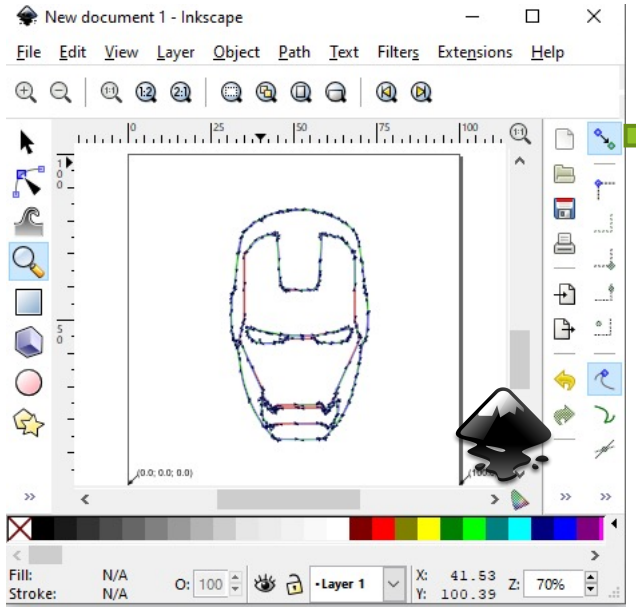
Outputs

```
for x = 0
  tag          values
( 0 0 )    [1498, 6760, 5.94067269606552, -0.003133998255179904, -1873, 2218]
( 0 1 )    [1516, 6761, 8.727153606580218]
( 0 2 )    [1529, 6762, 10.758482939351367]
( 0 3 )    [1540, 6763, 12.487833538186955]
( 0 4 )    [1549, 6764, 13.915182118840786]
( 0 5 )    [1558, 6765, 15.342656447749663]
( 0 6 )    [1566, 6766, 16.61917342351736]
( 0 7 )    [1573, 6767, 17.74472449379713]
( 0 8 )    [1580, 6768, 18.87037413194015]
( 0 9 )    [1586, 6769, 19.845045011140623]
```

```
Working area dimensions : 533.8192779049064 mm  *  266.9112088725073 mm
```

Results (Code-2)

Image Input

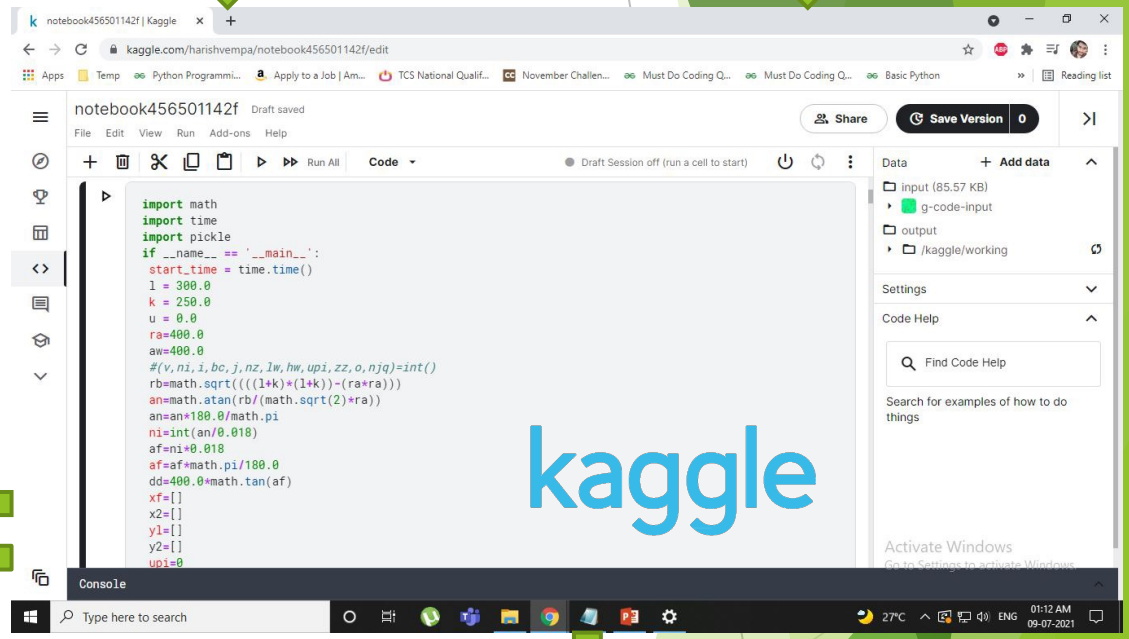


Gcode

```
G01 X64.685354 Y91.178443 Z-1.000000
G03 X63.341876 Y91.107433 Z-1.000000 I-0.400783 J-5.161893
G03 X63.273962 Y90.917012 Z-1.000000 I0.021278 J-0.114910
G02 X63.496924 Y90.471635 Z-1.000000 I-0.804890 J-0.681437
```

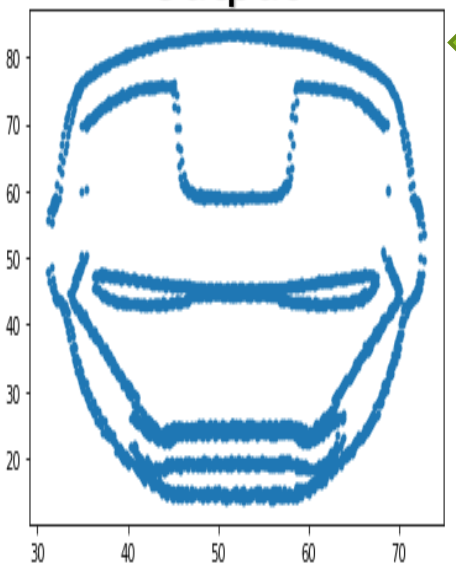
Code-1 Output

```
for x = 0
  tag        values
( 0 0 )    [1498, 6760, 5.94067269606552, -
( 0 1 )    [1516, 6761, 8.727153606580218]
( 0 2 )    [1529, 6762, 10.758482830351363]
( 0 3 )    [1540, 6763, 12.[illegible]]
( 0 4 )    [1549, 6764, 13.915182118840786]
( 0 5 )    [1558, 6765, 15.342656447749663]
( 0 6 )    [1566, 6766, 16.61917342351736]
( 0 7 )    [1573, 6767, 17.74472449379713]
( 0 8 )    [1580, 6768, 18.87037413194015]
( 0 9 )    [1586, 6769, 19.845045011140623]
```




```
import math
import time
import pickle
if __name__ == '__main__':
 start_time = time.time()
 l = 300.0
 k = 250.0
 u = 0.0
 ra=400.0
 aw=400.0
 #(v,ni,i,bc,j,nz,lw,hw,upi,zz,o,njq)=int()
 rb=math.sqrt((((l+k)*(l+k))-(ra*ra)))
 an=math.atan(rb/(math.sqrt(2)*ra))
 an=an*180.0/math.pi
 ni=int(an/0.018)
 af=ni*0.018
 af=af*math.pi/180.0
 dd=400.0*math.tan(af)
 xf=[]
 x2=[]
 yl=[]
 y2=[]
 upi=0
```

Simulated Output



Motor Driver code

```
[[-3216, 3746, -1616], [2, 2, -1], [2, 2, -1], [3, 2, -1],
1, -1], [1, 2, -1], [2, 2, -1], [2, 2, -1], [2, 2, -1], [1
-1], [1, 2, -1], [1, 2, -1], [2, 2, -1], [1, 2, -1], [1, 2
1], [1, 2, -1], [1, 2, -1], [1, 2, -1], [1, 2, -1], [1, 2,
[1, 2, -1], [0, 2, -1], [1, 2, -1], [0, 2, -1], [6, 3, -1]
2, -1], [5, 3, -1], [0, 2, -1], [5, 3, -1], [0, 2, -1], [5
1], [4, 3, -1], [4, 3, -1], [7, 4, -1], [3, 3, -1], [7, 4,
[3, 3, -1], [-1, 2, -1], [0, 2, -1], [2, 3, -1], [0, 2, -1
19, 0], [9, 2, 1], [0, 0, 0], [5, 0, 1], [5, 0, 1], [0, 0,
[20, 8, 1], [14, 5, 1], [11, 4, 1], [10, 3, 1], [8, 3, 1],
```

Coordinates plotted

```
[[44.35409926830525, 14.439183000709091], [44.20509357692208, 14.85481292937908],
[43.87506608732727, 15.830446267875077], [43.70282665670314, 14.743437542217356],
3.38173826816606, 15.719563900626204], [43.20078069749718, 16.13592297669182], [4
87986276574401, 15.608596473366049], [42.70788168158913, 15.881109836696954], [42
7821361744645, 16.714685303227927], [42.20634944402528, 17.1317178374075], [42.05
241401054, 16.604180944596788], [41.71410672309182, 16.87721487186113], [41.56543
28000584, 17.567875249816066], [41.21310182236289, 17.985552316517015], [41.06448
03322, 18.532722899531866], [40.72152961291485, 18.95083405942489], [40.563779393
671, 19.498717013651586], [40.22106565652135, 19.772822889083226], [40.072588046
```

Future Scope

Robotics has got a wide range of scope in every possible sector.

In our project, we have faced an issue, the end cell size is getting increased. Because, we have considered the x- axis coordinates based on stepper motor step angle.

Since, the cell size is approximately 0.0001mm it doesn't make much difference. But, in order to increase accuracy, we can work on adding a feedback element and error correction code.

CONCLUSION

- Proposed model approach looks at the problem of motor driver code generation in a new angle, which significantly reduces the code complexity.
- Robotics soon take a major role in various manufacturing and service sectors.
- Future deals with robotics!