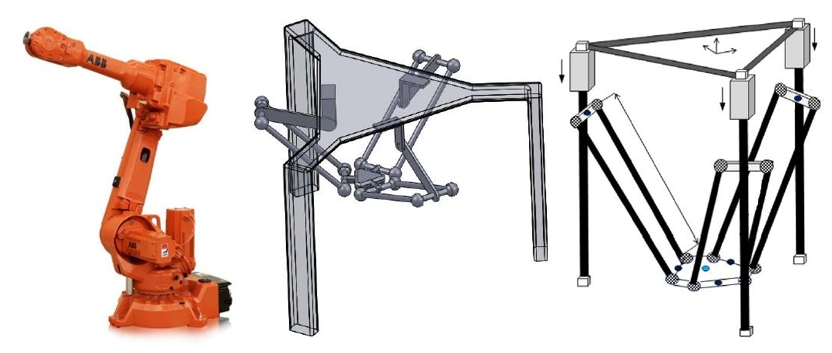
# Introduction

Robots are important in many industries, but they are different with other machines. For example, a single CNC machine gives high performance for a specific machining task, while single robot can achieve many tasks with a different performance for each because to the customized end effector. Robots are good at performing tasks that are too repetitive or dangerous for human to do such as welding, assembly and CNC operations.

Robots can be classified according to kinematic structure. A robot is said to be a serial robot or serial (open-loop) manipulator if its kinematic structure takes the form of an open loop-chain, a parallel manipulator (figure 1) if it is made of a closed-loop chain (Pandilov & Dukovski, 2014). Parallel robots have such advantages as high speed and acceleration, high stiffness and high accuracy. However, the drawbacks are also obvious such as low workspace/robot size ration, complicate calibration. As considering the features of both robots, serial robots are widely used on assembly line and parallel robots take the role in picking goods on conveyer belts.



**Figure 1.** Serial robot (left), parallel robot (middle) and linear delta robot (right) (Pandilov & Dukovski, 2014)

Delta robots are those parallel robots with three mechanism chains and it is believed that the first delta robot is designed by Clavel who uses parallelograms mechanisms and moving platform to create a robot with three translation and one rotation degree of freedom with respects to the base (Clavel, 1990). The application of delta robots begins in packaging industry in 1987, and it became one of the most successful commercial robots; in 1999 ABB Flexible Automation launched their delta robot “FlexPicker” which aimed at food, pharmaceutical and electronics industry (Bonev, 2001). Recently, parallel robots have found a place in households in the form of 3D printing which is called delta 3D printer, such products bring convenience to the creators for building plastic prototypes or doing experiments. For most of the mass production delta 3D printers, the end effector is driven by three linear actuators, as the result, they are also called linear delta robots.

The delta robots have attracted great interests not only in industry but also in academia. Variants of modified delta robots are built in university labs, for example, University of Maryland, University of Genoa and University of Western Australia (Bonev, 2001). At the same time, varies of researches have been carried out on robot workspace, dynamics and kinematics; methods of improving accuracy and performance are also analyzed. Generally, the method of researching a robot includes two essential steps. The first step is to analyze the kinematics equations which includes forward kinematics and inverse kinematics. Then the maximum workspace can be defined by numerical method or setting limits on the kinematics equations. Meanwhile, trajectory generation is another research direction, which focus on the velocity, acceleration, jerk for each degree of freedom.

The drawbacks of delta robots are obvious by including current researches, which is the kinematics problem of a parallel robot is (Liu, et al., 2003) complex. It does not mean that the kinematics cannot be solved, but hard to be solved with numerical method with high efficiency. The main reason is that the end effector has to be connected to the base with at least two legs. For such reasons, delta robots have the disadvantages in terms of relatively small useful workspace. According to (Craig, 2005)’s explanation, the workspace is one of the most important criteria of evaluating a robot’s performance, because it determines the area that the end effector can reach.

The linear delta robot has different kinematics with those invented by Clavel, but almost all the current products are designed with an enclosed shape (Isosceles right triangle) which leads to a problem that the size of the robot is larger than the maximum workspace. But one fact has been ignored by many studies, the motion of one single chain can be seen as transferring horizontal motion into vertical direction, which means delta robots can be designed foldable.

However, there is still no design used the advantage of the foldable structure. So, it is necessary to find out whether a foldable structure is feasible on linear delta robots to increase their workspace and robot size proportion. Nevertheless, the home 3D printer customers are complained about the unstable control system which leads to part failure and material wastes, so it is also necessary to research on the control algorithm, hardware chosen, manufacturing method, commanding rules and motor specifications to find out whether these aspects affect the usage of linear delta robots.

## Project Aims

Based on the problems that presented above, it is clear that the function of a domestic parallel robot is rare (only 3D printing), and researching on the mechanism and control algorithm of the delta robot is valuable. And on that conclusion the aim of doing this project is defined as to solve the problems that limit the usage of linear delta robots by improving the robot structure and control method.

In order to achieve the aim a new robot will be designed and the prototype of the new design will be manufactured and assembled; besides, the control algorithm will be programmed and tested on a microcontroller. The main points include analysing the relationship between robot specifications and its workspace; the robot structure designing and embedded system programing.

## Project Objectives

According to the project aim explained above, it is clear that this project can be divided into two parts, one is to design the structure, another is to design the control system. And there are five objectives.

Objective 1: Analyse the robot workspace

This objective is to build the theoretical model of the robot and determine the robot parameters.

Workspace is an important criterial to distinguish between a CNC machine and a robot. The purpose of workspace analysis is to determine the robot parameters. The processes include defining customer’s needs on workspace shape, robot size,

Objective 2: Design the linear delta robot

This objective is to design the mechanism of the robot based on theoretical models. There are always varies of designs come from one theoretical model, by analyzing comparing and evaluating these designs to define a balance solution. So variant models will be designed and analyzed, and an improved one will be manufactured and assembled as the prototype.

Objective 3: Design the control system on a micro controller

This objective is to design a fundamental structure of the control software. Such structure should show how the software works, and the functions needed. Before the design it is necessary to select a proper microcontroller, because useful hardware resources may make this process straight forward.

Objective 4: Design the trajectory planning algorithm

Trajectory describes the position, velocity and acceleration of each point on a path. The trajectory planning algorithm is to convert user’s command to a trajectory. A lot of equations and calculations will be presented during this objective.

Objective 5: Design the motor control algorithm

The motor control algorithm is to adjust the motor speed and acceleration. There will be some calculations to determine the relationships between motor parameters and physics dimensions. But the main focus of this objective is on the principle of driving motors with the hardware.

For the rest of this report, the researches that have been done will be presented as a literature review. The literature review will focus on what method or theory has been used in another research or design. Then, A chapter that explained what method used specifically in this project will be presented. After that, the design outcomes and some tests results will be shown. The results will be discussed in chapter 5, as well as the effect and performance of the method selected. Finally, this report will end with a conclusion.

# Research and Literature Review

This section is aimed to research on current 3D printers, in terms of the mechanism structure, user interface, manufacture method, kinematics and control algorithm. The aim of the research is to provide reliable technical methods to achieve the project aim.

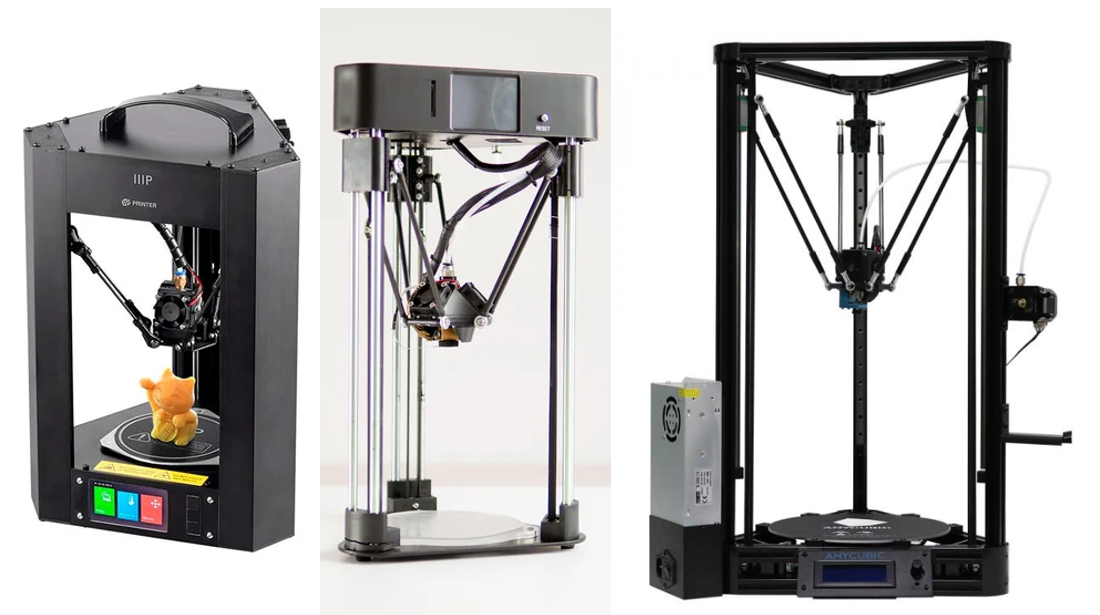
## Current 3D printers

With the development of low-cost micro controller and fused deposition technology, sales of home 3d printers are rising year by year. At the same time, home 3D printers with delta robot structure have a certain share in the market, according to the data from Amazon, 3D printers priced under $ 300 have higher sales, from which the three most sold printers are listed in table 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3D Printer | Print Area (mm) | Min. Layer Height (mm) | Max. Print Speed (mm/s) | Materials | Market Price |
| Monoprice Mini Delta | ø110 x 120 | 0.05 | 150 | 1.75mm ABS, PLA | $159.99 |
| BIQU Magician | ø100 x 150 | 0.1 | 70 | 1.75mm ABS, PLA | $169 |
| Anycubic Kossel Linear Plus | ø230 x 300 | 0.1 | 60 | PLA, ABS, HIPS, Wood | $200 |

**Table 1.** Specifications of current home delta 3D printers (Mensley, 2020)

According to the table, the resolution of most of the robots on the vertical axis is 0.1mm, the maximum printing speed is about 60mm/s to 70 mm/s, and all of these printers can print ABS and PLA materials. The structure of Delta 3D printers can be divided according to the motor position; an upper motor printer is shown in figure 2(b), the motor, controller and power supply are placed at the top of the printer. The lower motor printer is shown in figure 2(a)(c), the motors, hod bed and controller is placed at the bottom of the printer while the power supply is placed beside the printer.



**Figure 2.** Monoprice Mini Delta (left), BIQU Magician (middle) and Anycubic Kossel Linear Plus (right) (Mensley, 2020)

Most of the printer manufactures use NEMA 17 stepper motor, Arduino Mega controller and RAMPS 1.4 combination with A4988 stepper drivers. One reason is that opensource control firmware such as Marlin GRBL and Klipper can be used directly on those hardware. The functions of the firmware are:

1. Interprets g-code into motion, controlling the motors

2. Takes notice when limit switches are activated

3. Send response when commands are executed or a problem occurs

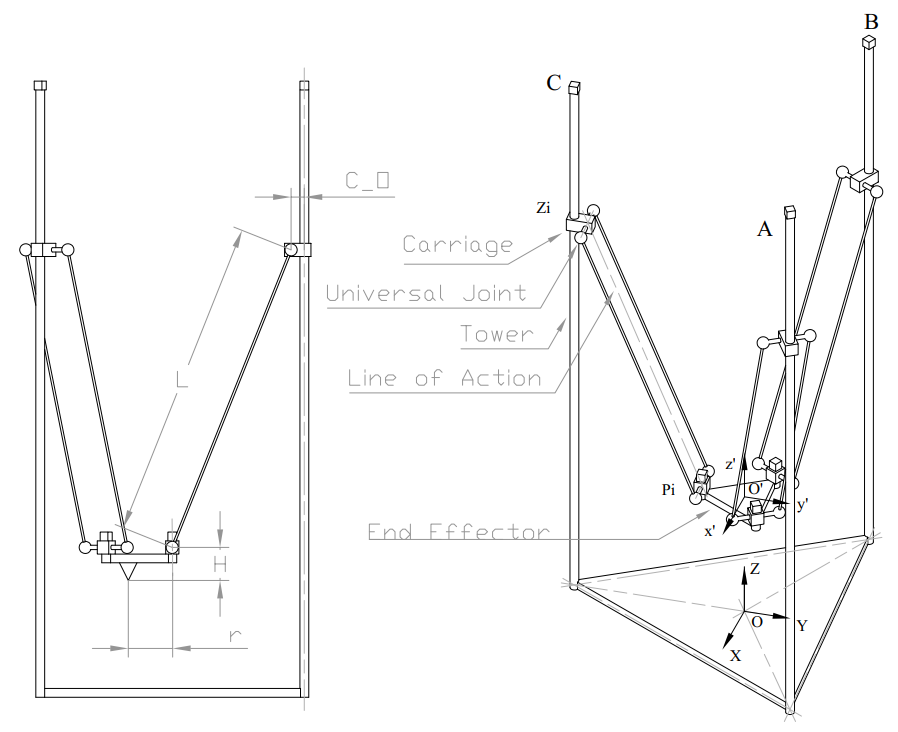
Which can be summarized as accepting inputs, driving actuators and sending feedbacks.

However, in the open source community such as Openedv and Stackoverflow, hobbyist found a lot of alternatives to improve the printer performance.

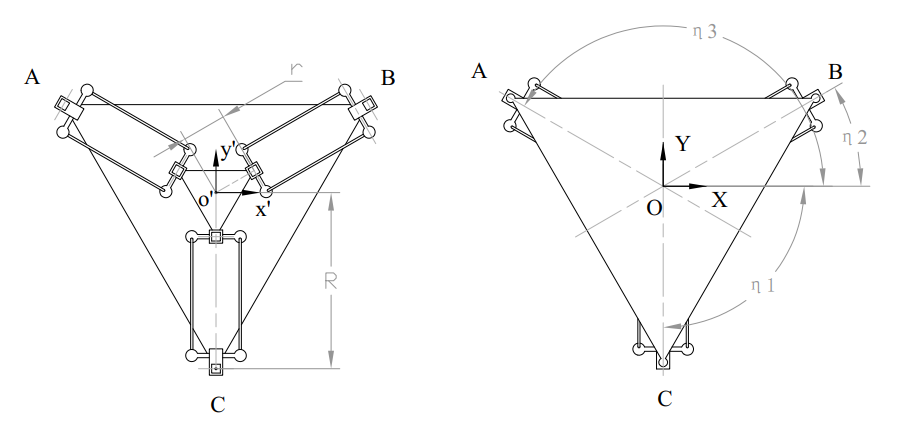
## Robot Structure

Generally, the relationships between the end effector and carriages can be determined by solving the kinematics, but parameters might be different if the tool or part size is changed. This section is to research on the configuration file of Marlin firmware to find out the parameters needed to define the kinematics on a real product.

The CAD drawing of a typical linear delta robot is shown in figure 3. The robot consists of three towers which is labeled as A, B and C; the slider labeled Bi on each tower is called the carriage, rods, carriages and the end effector are connected with universal joints. The dotted line between two rods represents the ideal position of a single mechanical chain, which will be used in workspace calculation.



**Figure 3**. CAD drawing of a linear delta robot



**Figure 4.** Top view of the CAD drawing of a linear delta robot

The robot can be described by 8 variables. L represents DELTA\_DIAGONAL\_ROD in Marlin, which means the centre-to-centre distance of the holes in the diagonal rods, generally the length of line of action equals to L; H represents DELTA\_TOWER\_OFFSET which is the horizontal offset from the middle of the robot to tower centre; r represents DELTA\_EFFECTOR\_OFFSET which is the horizontal offset of the universal joints on the end effector; H represents DELTA\_EFFECTOR\_HEIGHT, which is the vertical distance from the universal joints to the bottom of the end effector; C\_O represents DELTA\_CARRIAGE\_OFFSET, which is the horizontal offset of the universal joints on the carriages; ηi represents DELTA\_TOWER\_ANGLE\_TRIM, which is a set of angles that measured in degrees anticlockwise looking from above the robot (Graves, n.d.). All the variables mentioned above will be defined as macros and placed in the configuration head file.

## Choice of microcontroller

As is explained in section 2.1, most of the home 3D printers are equipped with 8-bit AVR mega 2560 micro controller (Arduino Mega). However, the Marlin firmware also support 32-bit controllers such as STM32 series, so it is necessary to research on different types of controllers to before the selection.

Basically, the controller performance not only infects printing speed but also infects the tolerance of code optimization. The STM32 is seems as an alternative of the Arduino Mega, according to table 2.3, the price difference between the two controllers is not large, IOs, timers and PWM ports of both controllers are the same.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | IOs | Timers | EXTI | PWM Ports | FPU | Cost |
| Mega 2560 | 100 | 14 | 6 | 8 bit 15 channels | N | ¥308.00 |
| STM32F407 | 100 | 14 | 6 | 16 bit 24 channels | Y | ¥135.00 |
| STM32F103 |  |  |  |  | N | ¥95.00 |

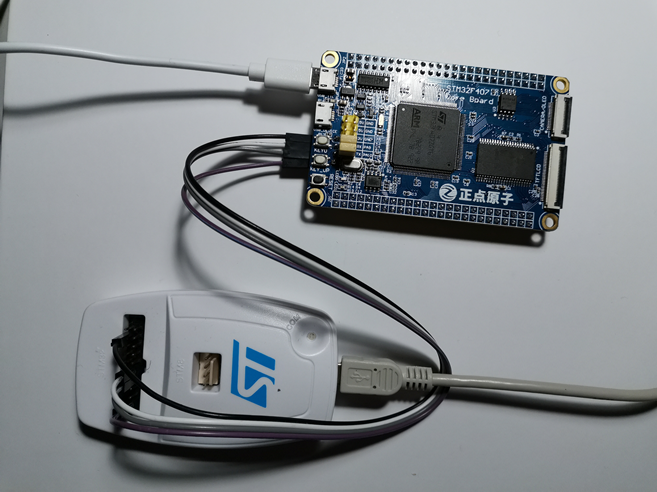
**Table 2.** Comparison between microcontrollers

https://detail.tmall.com/item.htm?spm=a230r.1.14.23.1ba249256ZSkQn&id=38816270178&ns=1&abbucket=6

<https://zhengdianyuanzi.tmall.com/?ali_refid=a3_430582_1006:1267360122:N:9nyPlE3iv4IpeEvXe%2FK1RseHVtKo2a5g:71f128fb398e1580803f5a0386142448&ali_trackid=1_71f128fb398e1580803f5a0386142448&spm=a230r.1.14.4>

AVR mega 2560 is an 8-bit micro controller that supports the Arduino library so that register operations can be replaced with simple functions, however, overly streamlined function library makes functions single. The reason of using Arduino is that the skills necessary to create an automated scientific tool were often not worth the investment of a researcher in a much different field (Pearce, 2014),.

STM32 is a 32-bit commercial controller, which can be programmed with either register operation, standard library or Hal library. Thanks to the two electronics company Alientek and Embedfire, free teaching materials and data manuals can be found in the open source communities, which brings convenience to novices. Besides, debug probes such as ST-Link and J-Link provides run-time variable monitor software such as STM Studio and J-Scope so that the output can be plot on a scope.



**Figure 5.** STM32F407 controller with ST-Link V2 debugger

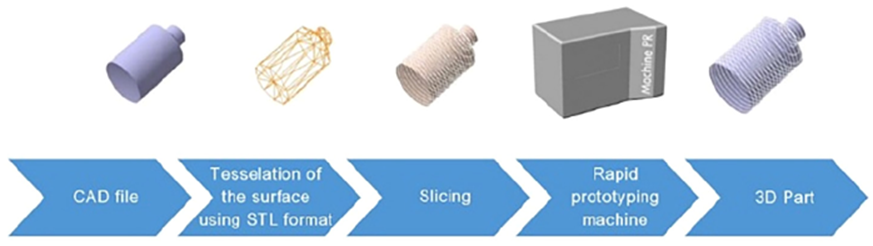
The main difference between STM32F4 series and STM32F1 series is that the new cortex M4 chip is embedded with a float point unit (FPU). According to (Barr & Massa, n.d.), manipulating float is only recommended on those controllers with FPU, because the compiler-supplied floating-point library contains a set of software subroutines which is recognized as IEEE 754 standard that emulate the instruction set of a floating-point coprocessor. However, there is still a solution, which is to reimplement the calculations themselves using fixed-point arithmetic only. The decimal fraction can be presented as integer by multiplying an integer such as 10 or 100.

## Manufacture Method

Production models use Aluminum rods as the structure, and injection molding plastic parts as joints. However, injection molding is costly and not suitable for building protypes, an alternative is to use 3D print parts, such idea comes up with the RepRap project (Bowyer, 2005). The RepRap project has created 3-D printing machines that can manufacture approximately half of their own parts (57% self-replicating potential excluding fasteners, bolts and nuts) from sequential fused deposition of a range of polymers and common hardware (Pearce, 2014). Another reason of researching on 3D print technology is that most of the linear delta robots are used for 3D printing.

### AM and 3D print

The term 3D printing is also called additive manufacturing (AM), and it is the construction of a three-dimensional object from a CAD model or a digital 3D model (Anon., 2013). Such method is believed to reform the commercial manufacturing sector especially the polymer industry, because AM does not have molding defects such as melt line and weld line (Rao & Harshitha, 2019). The process of 3D printing includes four steps from figure 6, where the input is a CAD file. The 3D part is produced layer by layer directly from a 3D CAD model through steps like designing using software, slicing and other processes.



**Figure 6**. Processes of 3D printing (Rao & Harshitha, 2019)

### Fused deposition modelling

Fused deposition modelling (FDM) is one of the 3D printing methods which is commonly used for thermal plastic material. Meanwhile, FDM is also used on most of the home 3D printers (those use Marlin control firmware). The principle of FDM is to extrude the thermoplastic filament through a circular nozzle to build 3D objects layer by layer (Rao & Harshitha, 2019). As a result, FDM is easy to handle and cost-efficiency.

### Material selection

Current researches have already done a lot of experiments to define the properties of 3D printed parts, which make it easier to do material selection and mechanical design. PolyLactic Acid (PLA) and Acrylonitrile Butadiene Styrene (ABS) are the two materials that are widely used on home 3D printers and their cost and properties are listed in table 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cost | Tensile Strength | Flexural modulus | Elongation |
| PLA | ¥89/kg | 37MPa | 4GPa | 6% |
| ABS | ¥69/kg | 27MPa | 2.1-7.6GPa | 3.5-50% |

**Table 3.** features of 3D printer materials

One experiment done by (Rao & Harshitha, 2019) pointed out that an ISO standard bolt that is printed with PLA has a larger shear stress and equivalent stress than the one printed with ABS. Besides, the PLA bolt can stand a larger force before deficiency occurs. All the results that from ANASYS software simulation proved that material ABS is weaker than PLA material.

However, the manufacturing process affects the material properties. According to (Lay, et al., 2019), FDM printed polymers are much weaker than the those are injection mould. For example, the tensile strength of FDM printed PLA part is 48% lower than the injection mould part, and the FDM printed ABS part has a lower Young’s modulus and tensile strength as comparing with the injection mould part.

## Current Method of Commanding

As is described in section 2.4.1, the input CAD file should be transferred to STL formal and sliced before printing. One of the slicing software is Ultimaker Cura, which prepare prints with a few clicks, integrate with CAD software for an easier workflow, or dive into custom settings for in-depth control (Braam, 2014). The output of the slicing software is a set of commands that can be read by the 3D printer control firmware through serial port or SD card. In other words, the control firmware is determined by the commands.

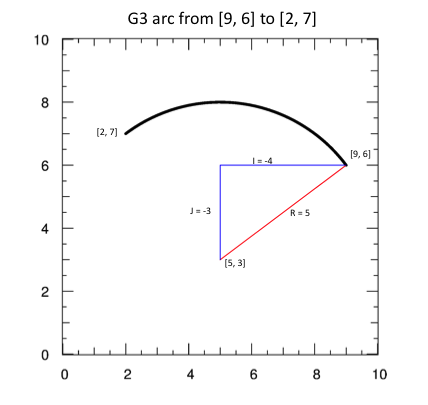
Most of the current firmware such as Marlin, GRBL and Klipper use G-code as the input. Current G-code standard is also known as RS-274 which was settled by the Electronic Industries Alliance in the early 1960s. Such standard specified the two basic motions: Linear motion and Arc motion (Kramer, et al., 2000).

### Linear motion command

Linear motions are often labelled as G0 or G1. nominally, each axis moves at constant speed and all axes move from their starting positions to their end positions at the same time. Coordinated linear motion can be performed either at the prevailing feed rate, or at traverse rate. If physical limits on axis speed make the desired rate unobtainable, all axes are slowed to maintain the desired path. The difference between G0 and G1 is different in different firmware. For example, both commands have the variable of feed rate in Marlin, and end effector variable such as printer extrusion rate can only be set with G1; however, the feed rate of G0 is a constant in GRBL firmware (Skogsrud, 2009).

### Arc motion command

Arc motions are labelled as G2 or G3. Where G2 represents clockwise move and G3 represents counter-clockwise move. However, before the motion is executed the workspace plane XY, ZX, or YZ should be selected. As a result, the arc motion command only needs two variables for each coordinate, and the third parameter is quote from the start point. An example is shown in figure 7 which illustrates a counter clockwise arc, starting at [9, 6]. It can be generated either by G3 X2 Y7 I-4 J-3 or G3 X2 Y7 R5.



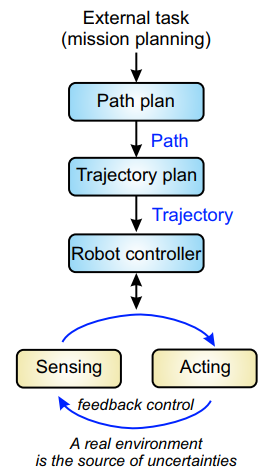
**Figure 7**. the G3 arc command

### Command format

The RS-274 standard specifies the G-code format, and the interpreter should signal an error if the command is illegal. Some characters such as spaces and tabs can be used to make the command more understandable, because they will be ignored on a line of command, and they do not change the command meaning, meanwhile, blank line can also be ignored.

## Current method of planning trajectory

G-code commands describe the path of the end effector, but in robotics, the term trajectory is more commonly used. The difference between path and trajectory is pointed out by (Hlaváč, n.d.) where the path provides a pure geometric description of motion and trajectory can be regarded as a path plus velocity and acceleration. According to (Craig, 2005) trajectory is described as the desired motion of a manipulator in multidimensional space, and in robot design it refers to a time history of position, velocity, and acceleration for each degree of freedom.



**Figure 8**. flow chart of planning robot trajectory (Hlaváč, n.d.)

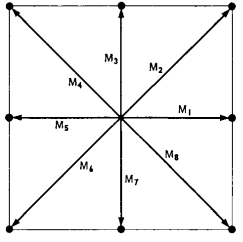
### Joint scheme and Cartesian scheme

Generally, there are two schemes to generate a path. Joint scheme is to describe the path shapes in terms of functions of joint angles (Craig, 2005). Generally, this scheme is used on serial robots, because each of their joint is equipped with a servo motor, so using the joint scheme is much efficiency; another reason is that joint scheme can solve the problem of singularity because serial robots have long mechanism with multi joints.

Another scheme is called Cartesian scheme which is much easy to understand, however, Cartesian schemes are more computationally expensive to execute, because, at run time, inverse kinematics must be solved at the path-update rate (Craig, 2005).

### Bresenham’s algorithm

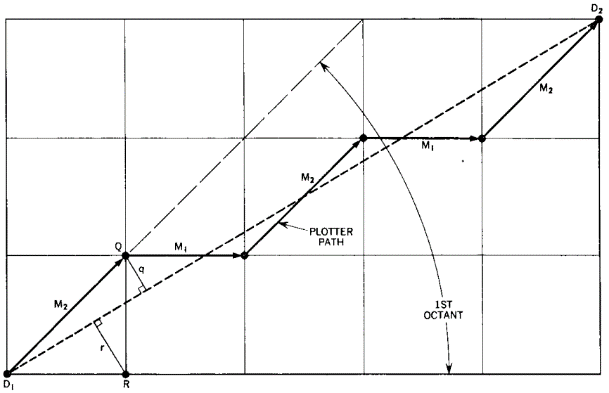
The path is a set of coordinates that described not only by a desired destination but also by some intermediate locations, or via points, through which the manipulator must pass enroute to the destination. When a G-code command is interpreted, a path is then generated. The criteria of evaluating path is smooth, however, the path cannot be as smooth as theoretically because of the motor properties which means the smooth path have to be approximated. The solution in GRBL is to used Bresenham's line algorithm to approximate a straight line into a set of points (Skogsrud, 2009).



**Figure 9**. eight linear movements from Bresenham’s algorithm (Bresenham, 1965)

Bresnham’s algorithm is commonly used on digital computers as a digital plotter (Bresenham, 1965). The fundamental idea is to simplified any of the motions into eight linear movements shown in figure 9. meanwhile the lengths of movements are determined by the resolution.

In most cases, the ideal path will pass between any of the two directions, then the question becomes to determine which standard movement is closer to the ideal one. For example, two adjacent data points and is shown in figure 10, since point is closer to the ideal path than , vector is chosen as the next movement.



**Figure 10**. Bresenham’s algorithm on linear motion (Bresenham, 1965)

### Sub movements

The Marlin firmware uses Bresenham’s line algorithm to generate the path too (Zalm, 2011). Moreover, the term block is used to describe a sub movement between the two points on the trajectory; therefore, the process of executing the control system is to read and execute the blocks in sequence. More details about the speed profile of the blocks will be presented in section 2.8.

## Current Method of Solving Kinematics problem

The kinematics problem of a linear delta robot can be divided into two parts the inverse kinematics (IK) and forward kinematics (FK) (Craig, 2005). IK problem is to convert the cartesian coordinate to the position of linear actuator, otherwise FK problem is to convert the linear actuator position to end effector position.

### Inverse kinematics

The Inverse kinematics problem of a linear delta robot is simple, which can be presented as a polynomial (Stock & Miller, 2003); (Liu, et al., 2003). A general method is to find a loop-closure of each limb that is holding with respect to the base and a typical inverse kinematics equation can be presented as equation (1).

|  |  |
| --- | --- |
|  | (1) |

Where x, y, z is the position of the end effector with respect to the base, L is the chain length, zi is the carriage position with respect to the base, and xi and yi are determined by the distance of each tower to the origin point and the shape of end effector.

### Jacobian matrix

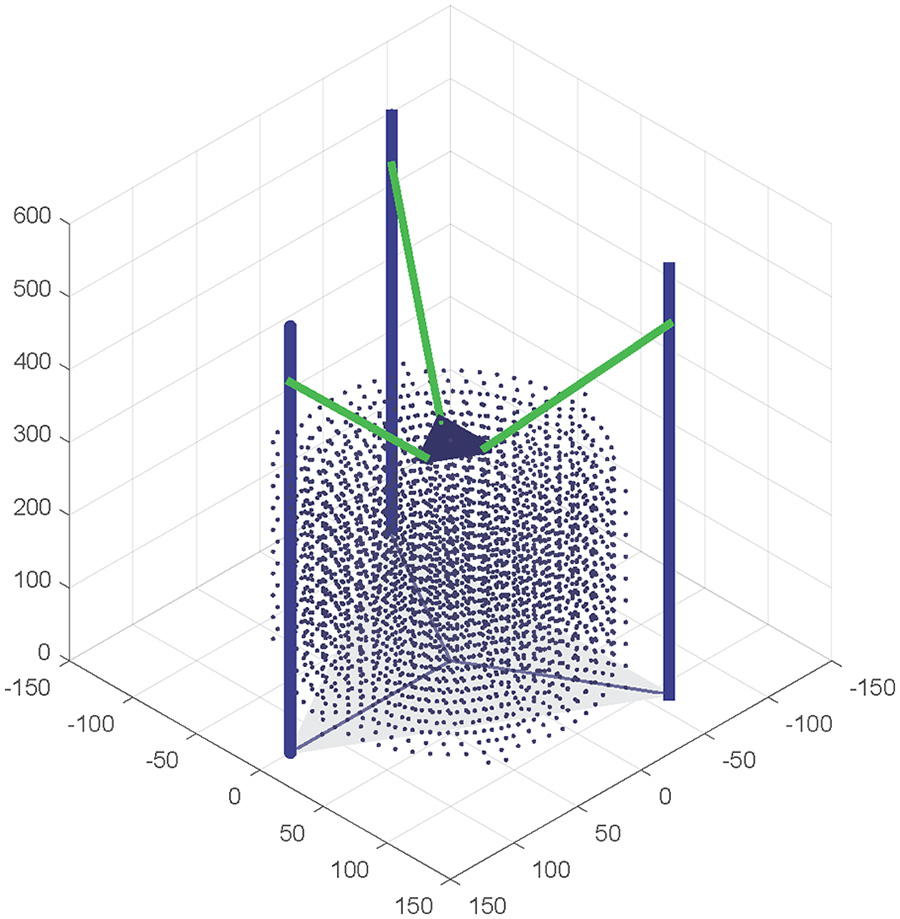
The Jacobian matrix is defined as the matrix that maps the relation between the velocity of the moving platform and the actuated joint rates (Stock & Miller, 2003). A typical Jacobian matrix can be presented as equation (2).

|  |  |
| --- | --- |
|  | (2) |

Where is the vector of end effector velocities and is the vector of carriage velocities and is the Jacobian matrix. The Jacobian matrix can only be defined after the inverse kinematics equation is applied.

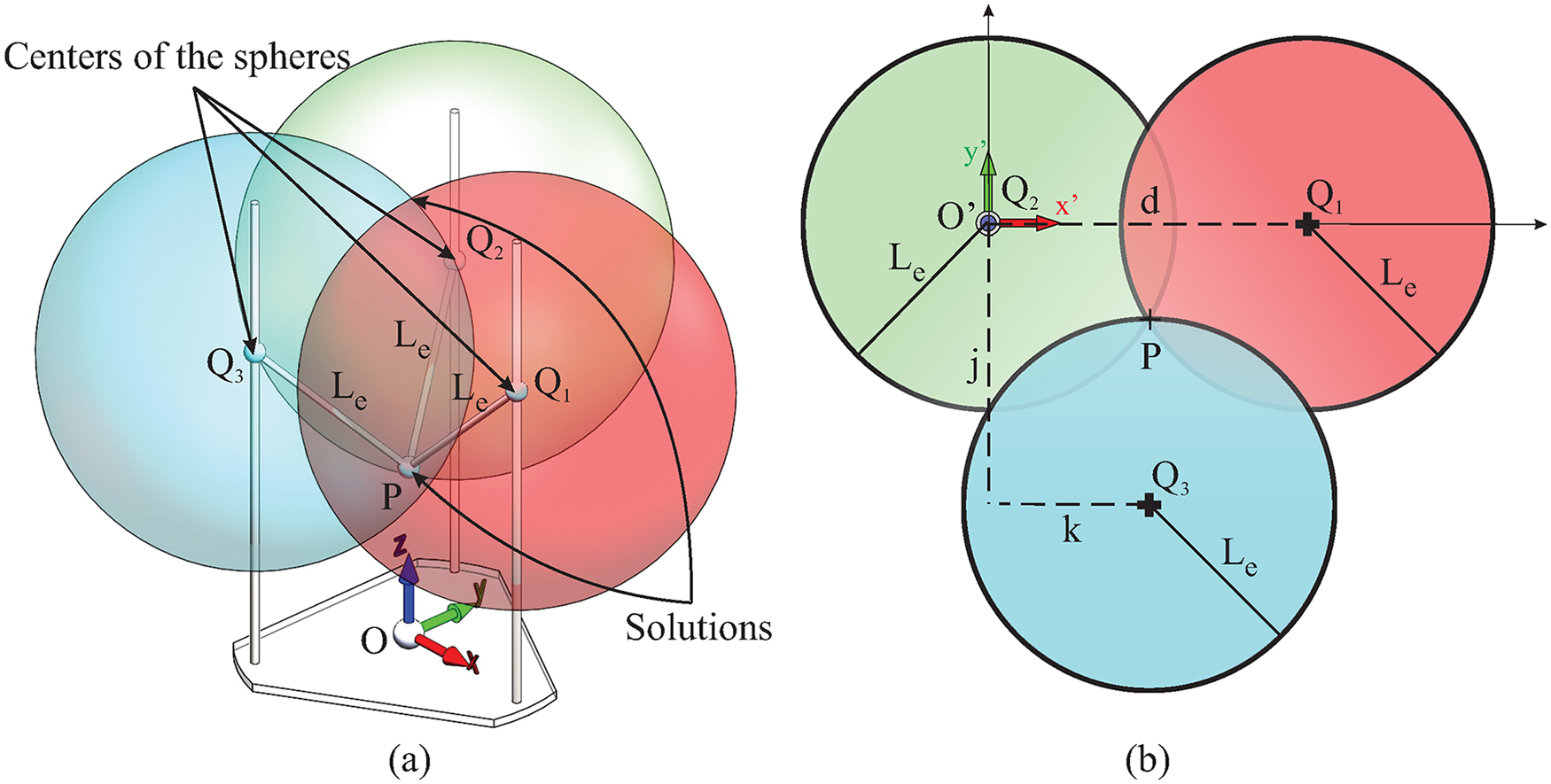
### Forward kinematics

Calculating FK in terms of polynomial has been proved by many researches such as (Liu, et al., 2003) and (Yuan, et al., 2008). It can also be applied with numerical method. However, FK problem is hard to solve directly, because the process is to solve ternary quadratic equations. Figure 11 is the simulation result with Matlab by (Rodriguez & Jaimes, 2019).



**Figure 11**. forward kinematics simulation with Matlab (Rodriguez & Jaimes, 2019)

The purpose of solving forward kinematics in Marlin firmware is to initialize the coordinate system and update the information that presented on LCD panel. The method of calculating FK in Marlin firmware is called trilateration (Zalm, 2011). The principle is that the coordinate of a point can be determined if its distance to other three certain points is curtained. This algorithm is also mentioned in (Rodriguez & Jaimes, 2019), in which the model can be represented as the intersection of three spheres as illustrated in figure 12. The centers of the spheres Qi are not the position of carriages but the offsets xi,yi,zi in equation (1). The intersections point P is the geometry center of the end effector, and the lengths between Qi and P are the same which is the length of mechanism chain.



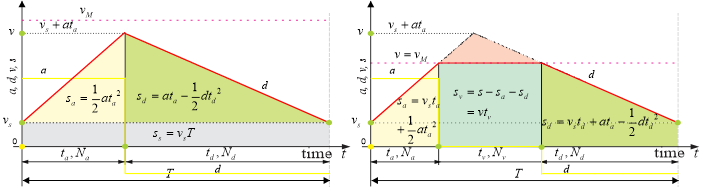
**Figure 12**. model of solving forward kinematics with trilateration (Rodriguez & Jaimes, 2019)

## Current stepper motor control algorithm

The trajectory is expected to be smooth, because rough, jerky motions tend to cause increased wear on the mechanism and cause vibrations by exciting resonances in the manipulator. A smooth function is defined as a function that is continuous and has a continuous first derivative (Craig, 2005). Generally, there are two schemes to make the trajectory smooth, one is s-cure profile and another is trapezoidal-curve profile.

Most of the designs such as those are listed in table 2 are equipped with NEMA 17 stepper motors without encoders, which means the control system is open looped, and a proper motor driving algorithm should be designed. Moreover, with the application of motor driver chips such as A4988 which has been mentioned before, the output of the controller is in forms of impulse. However, there is still a problem which is the impulse-time diagram does not match the velocity-time diagram.

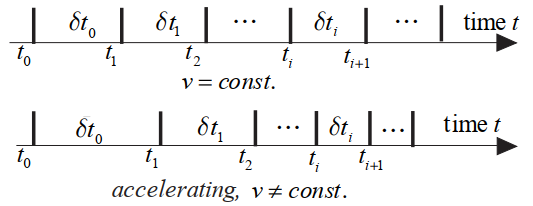
This problem is analyzed by (Stoychitch, 2018), and according to his research, the algorithm includes two parts: trajectory planning and pulses generating. The first step is to distinguish whether the motion is trapezoidal or triangular. Trapezoidal motion includes an acceleration part, a constant velocity part and a deceleration part, while the triangular motion has no constant velocity part. The triangular may occur when the total steps required is not enough for accelerating to meet the maximum velocity.



**Figure 13**. triangular (left) and trapezoidal (right) speed profile (Stoychitch, 2018)

According to (Stoychitch, 2018), the proportion between the steps amount in each section should be the same as it between the corresponding areas. At the same time, another problem comes which is the areas can be presented in terms of float or double precision float, but the step amount can only be presented in integer. In order to minimize the effect of losing the decimal part, steps for constant velocity motion should be first determined, and steps for acceleration should be determined after the deceleration steps are considered.

There are to schemes to design the pulse generating algorithm, one is the “time per step” schemes, another is the “steps per time” scheme. It has been proved that the “time per step” scheme is much suitable on processors, especially those with floating-point unit (FPU), while another one is suitable for FPGA. The purpose of the “time per step” scheme is to determine the time delay between two impulses which is shown in figure 14.



**Figure 14.** Constant speed and accelerating step sequences (Stoychitch, 2018)

## Stepper motor theory

It is clear that the speed profile is important in motor control algorithm, but the maximum acceleration, deceleration and revolution can only be defined after analyzing the motor specifications.

### Stepper motor torque

According to (New Japan Radio Cp.,Ltd, n.d.), the torque produced by a stepper motor depends on the step rate, drive current in the windings and the drive design or type. In order to define the relationship between the torque and speed characteristics, it is necessary to find out the following elements:

1. Holding torque

The amount of external torque which must be exerted on the shaft motor when the motor is at full rated current and is at rest (zero speed).

1. Pull-In Torque

The amount of torque in which that motor may move the load without acceleration.

1. Maximum Start Rate

The maximum starting step frequency with no lad applied

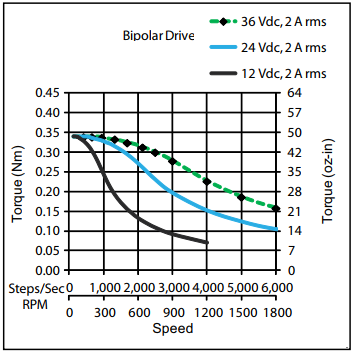
1. Pull-Out Torque

The highest torque a stepper motor can output at a given speed without losing steps.

1. Maximum Slew Rate

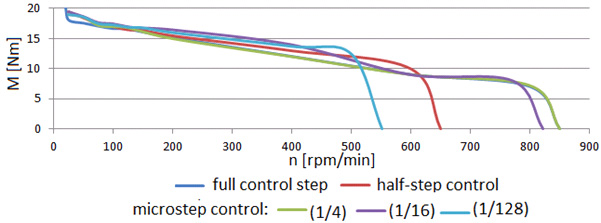
The maximum operating frequency of the motor with no load applied.

The torque profile of the motors can be found from its manufacture, for example, figure 15 shows the torque profile of NEMA 17 stepper motor from which the torque decreases when the speed increases, and the motor can hardly provide any torque when the speed reaches 4000 steps /sec if the power supply is 12 volt.



**Figure 15.** Torque profile of a NEMA 17 stepper motor (PBC Linear, n.d.)

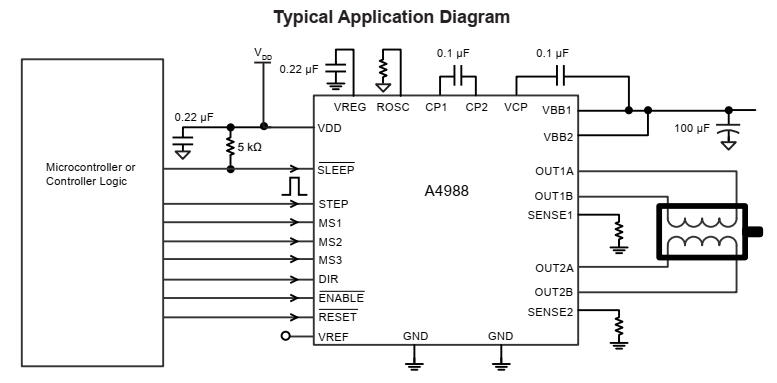
Moreover, a research points out that the torque profile is affected by microstepping resolution. As is shown in figure 16, the torque profile of full control step is similar to that of quarter control step, while half-step control mode can provide a high and steady torque until the motor speed meets 600rpm.



**Figure 16.** Torque profile of stepper motor in different resolution (Warguła, et al., 2017)

### Stepper motor driver

An application diagram of A4988 motor driver is shown in figure 17, there are nine pins needed to be connected to the microcontroller, four pins for stepper motor, and two pins to connect the motor power supply. However, according to the explanation from (Pololu Robotics & Electronics, n.d.), the sleep pin can connect with reset pin.

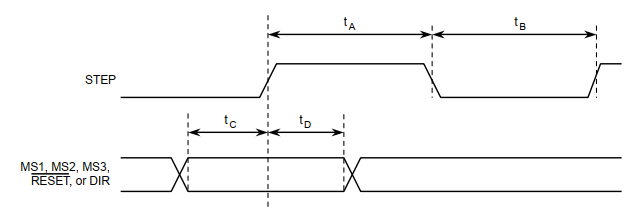


**Figure 17.** Typical application diagram of A4988 motor driver (Allegro MicroSystem, 2020)

The microstep select pin (MSx) are used to set the resolution by inputting the logic voltage, and the minimal microstep resolution is sixteenth step, a truth table of resolution is shown in table 4. The principle of drive the stepper motor with A4988 motor driver is to send pulse signal through the STEP pin, and the motor speed is related to the signal frequency. However, the width of each pulse should meet the demand of the driver, or they will never been detected. Figure 18 demonstrates the logic interface timing, from which the pulse width specifications are shown in table 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MS1 | MS2 | MS3 | Microstep Resolution | Excitation Mode |
| L | L | L | Full Step | 2 Phase |
| H | L | L | Half Step | 1-2 Phase |
| L | H | L | Quarter Step | W1-2 Phase |
| H | H | L | Eighth Step | 2W1-2 Phase |
| H | H | H | Sixteenth Step | 4W1-2 Phase |

**Table 4.** Microstepping resolution truth table (Allegro MicroSystem, 2020)



**Figure 18.** Logic interface timing diagram (Allegro MicroSystem, 2020)

|  |  |  |  |
| --- | --- | --- | --- |
| Time Duration | Symbol | Typ. | Unit |
| STEP minimum, HIGH pulse width |  | 1 | μs |
| STEP minimum, LOW pulse width |  | 1 | μs |
| Setup time, input change to STEP |  | 200 | ns |
| Hold time, input change to STEP |  | 200 | ns |

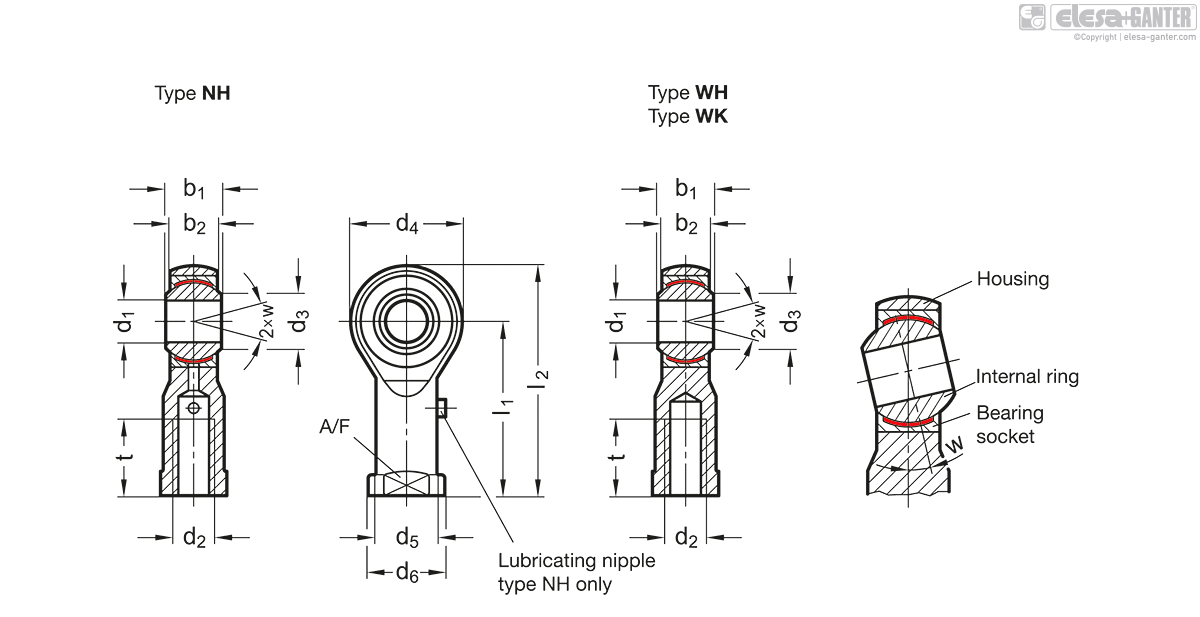
**Table 5.** Time duration for logic interface timing diagram (Allegro MicroSystem, 2020)

# Method

In this chapter the methods used to achieve the objectives are explained. The process of designing the robot begins with analysing the workspace which is seem as the requirement; then by study on and modify on current models a mature and conservation scheme can be determined; after that feasibility analysis will be carried on based on component limits, costs and failure mode. The control system can be designed as the mechanism is designed. A flow chart of the whole program will be given; each function will be presented in detail in both theories and programs.

## 3.1. Workspace analysis

Previous studies had already explained the general processes of analyzing the workspace of a linear delta robot, which is first define the forward kinematics equation and then add limits on it. However, in this section the analyzation begins with the joint limits, which is much closer to engineering problems. Ball joints are widely used on delta robots as the connections between rods and sliders because they can provide rotations around two perpendicular axes. However, one of the rotations is limited due to the physical structure, figure 19 is the drawing of a ball joint, from which the joint limit angle is presented as 2 \* w.



**Figure 19.** engineering drawing of ball joints

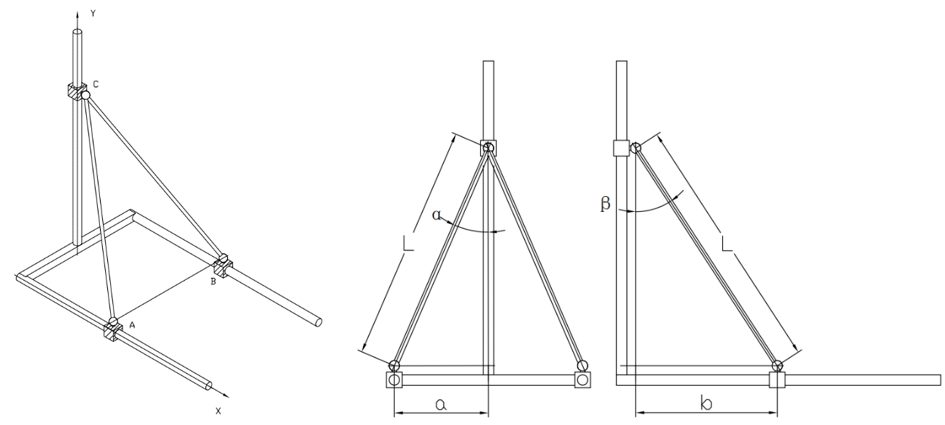
<https://www.elesa-ganter.com/en/www/Joints--Stainless-Steel-Ball-joint-heads-with-female-thread--GN6485>



**Figure 20.** Carriage and mechanism chain of Kossel linear plus

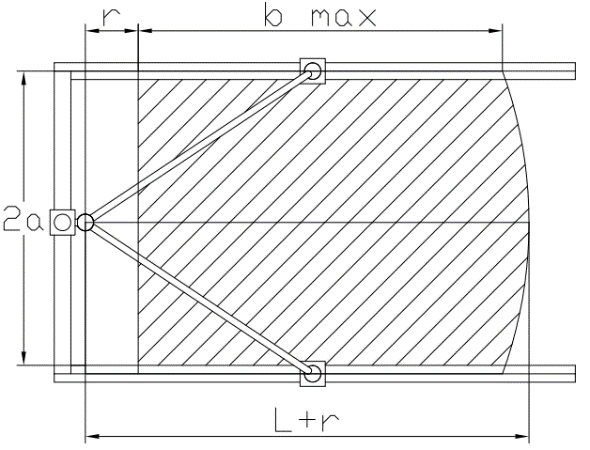
Suppose the unlimited rotation is around the horizontal axis, then the sketch is shown in figure 20. The method of analyzing the workspace is first to simplify each chain into a rod-slider system which is shown in figure 21, the two rod AC and CB represent the edge positions that the chain can achieve, the two rods are linked to three sliders with ball joints. Slider C represents the linear actuator that can move along y axis. Slider A and slider B can move along x axis, meanwhile line AB is perpendicular to both x and y axis, because each chain is a four-bar mechanism. The area that line AB swept is the work plane, which can be calculated as:

|  |  |
| --- | --- |
|  | (3) |



**Figure 21.** Model of defining workspace of a single chain

The limit angle of those ball joints that used widely on home 3D printer is about , and the range of angle is from to . The actual workspace of a single chain is shaded in figure 22, the arc at the end of the work plane only can be achieved when angle is nearly . As considering the radius of the end effector the workspace translates the length of the radius along the x axis.



**Figure 22.** Maximum workspace of a single chain

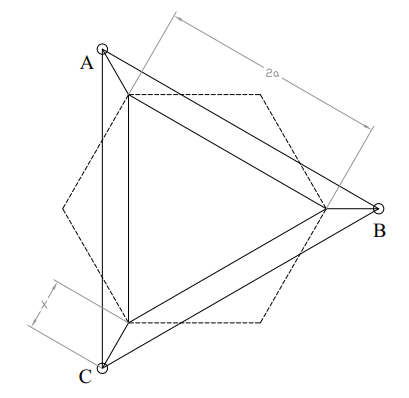
In conclusion, the features of the workspace can be summarized as the following points

1. Rectangular shape
2. End effector radius decreases the proportion between workspace and robot size
3. Longer chains create larger workspaces

Since the workspace of a single chain can be approximated as a rectangle, then the work comes to combine these three rectangles and find the largest covered area. Theoretically, there are three types of structures: equilateral triangle structure, obtuse triangle structure and Isosceles right triangle structure.

### Equilateral triangle structure

This structure is applied on most of commercial 3D printers, the frame shape is equilateral triangle and the shape of maximum workspace is hexagon. But in fact, the actual workspace (3D printer hot bed) is the inscribed circle of the hexagon as considering the manufacture cost.



**Figure 23.** Line graph of equilateral triangle structure

A linear graph of equilateral triangle structure is shown in figure 23, where A B and C are the position of carriage and the larger triangle represents the robot size. The inner triangle is the ideal robot size, and the offset between actual one and ideal one x represents geometry size of tower, carriage and shell. The relationships between actual robot size S and geometrical offset x on a constant workspace can be presented as equation 4.

|  |  |
| --- | --- |
|  | (4) |

Another design that based on this structure has been built by (Oberhauser, 2016) who let the limited rotation around the horizontal axis, which is shown in figure 24, where is the joint limit angle, is the offset angle, and is the width of the workspace concentric circles.

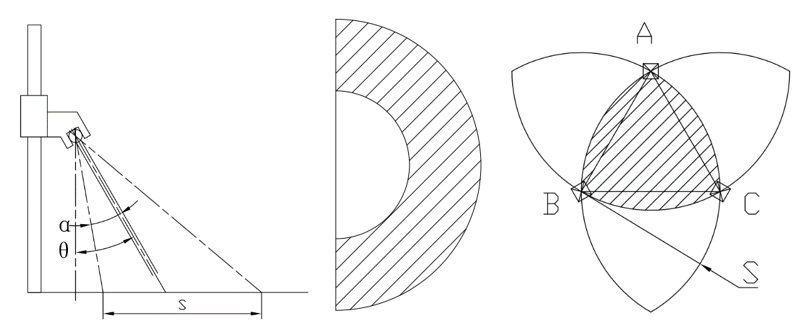


Figure 24. Side view of the mechanism chain (left), workspace of a single chain (middle), workspace of the robot built by (Oberhauser, 2016) (right)

The expression of distance S can be written as:

|  |  |
| --- | --- |
|  | (5) |

Where L is a constant.

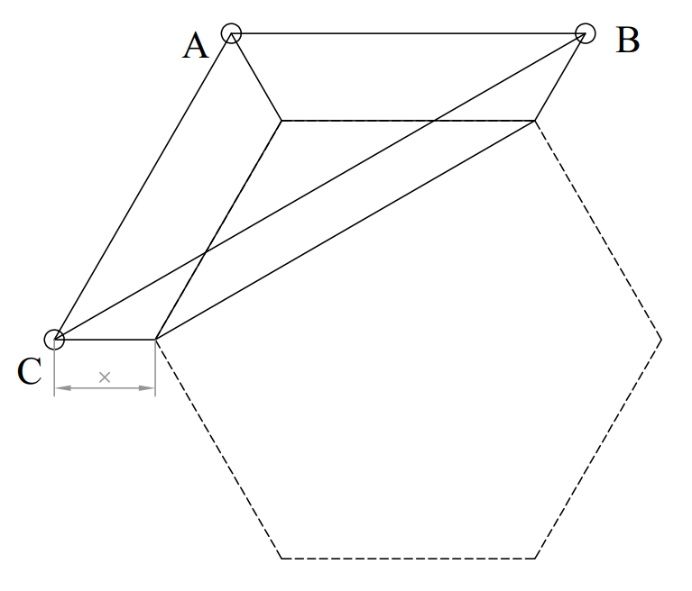
|  |  |
| --- | --- |
|  | (6) |

The equation has a lowest solution. In other word let p = 0 is the optimized solution, and the equation can be presented:

|  |  |
| --- | --- |
|  | (7) |

### Obtuse triangle structure

This structure has the same hexagon workspace as the previous one, the difference is that the direction between each motor is . As comparing figure 3.1.5 and 3.1.4, there is no adopt that the obtuse triangle structure is much smaller than the equilateral triangle one. However, such scheme has not been applied on commercial products, because it increases the size of the end effector dramatically.



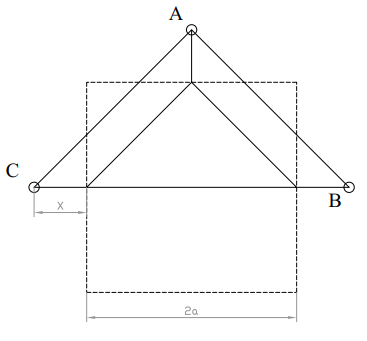
**Figure 25.** Line graph of obtuse triangle structure

The relationships between actual robot size S and geometrical offset x on a constant workspace can be presented as equation 8.

|  |  |
| --- | --- |
|  | (8) |

### Isosceles right triangle structure

This structure has a square workspace which is shown in figure 26, such model has been mentioned and proved in (Yuan, et al., 2008), but there is still no production model for commercial purposes. In fact, a square or rectangular workspace is as welcome as the circular one according to the sales of cartesian 3D printers. Another advantage is that the square workspace is suitable to materials such as paper and sheet material.



**Figure 26.** Line graph of isosceles right triangle structure

The relationships between actual robot size S and geometrical offset x on a constant workspace can be presented as equation 9.

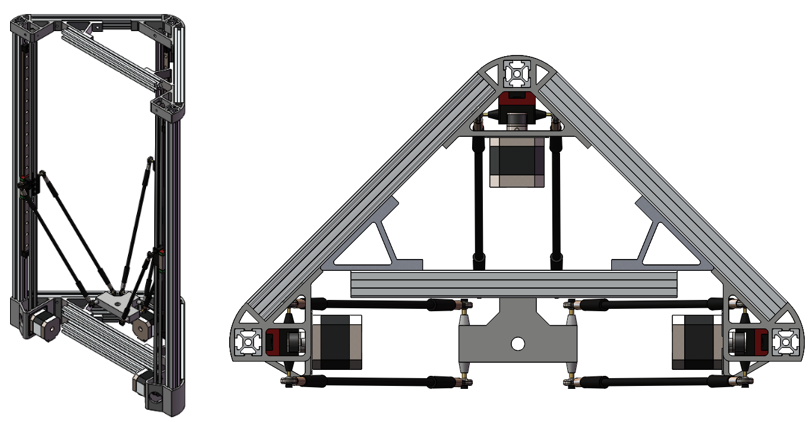
|  |  |
| --- | --- |
|  | (9) |

## Robot design

The robot will be designed with isosceles right triangle structure, because such structure is practical and has been theoretically proved. But the robot design process begins with doing modifications on current products.

### Modified Kossel linear plus

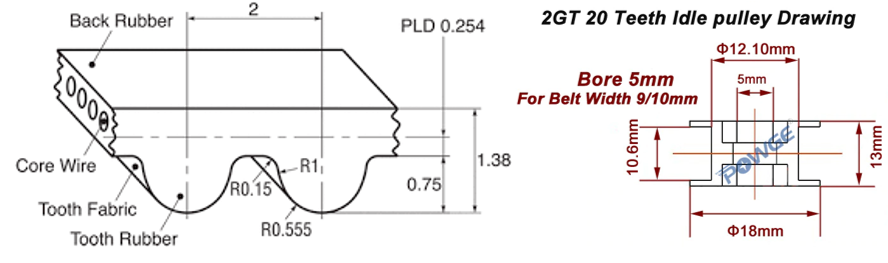
The first design is a modified Kossel linear plus 3D printer which is shown in figure 27. The robot has a square workspace but reserved almost all the features on the origin product. The three motors are placed at the bottom, and another plane is needed to cover over the motors, such design provides a stable structure and lower center of gravity.



**Figure 27.** Modified Kossel linear plus 3D printer

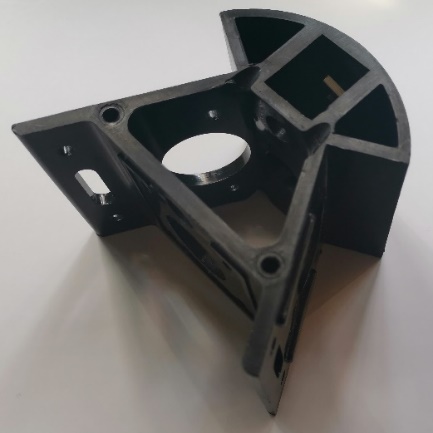
The transmission system uses of timing belts to actuate robot carriages vertically along a linear rod/rail as shown in Figure 20. The stepper motor is NEMA 17 which is a two-phase variable-reluctance stepper motor, the drawing of the 2GT timing belt and belt pully is shown in figure 28. The diameter is 13mm, as a result the length of each step can be calculated:

|  |  |
| --- | --- |
|  | (10) |



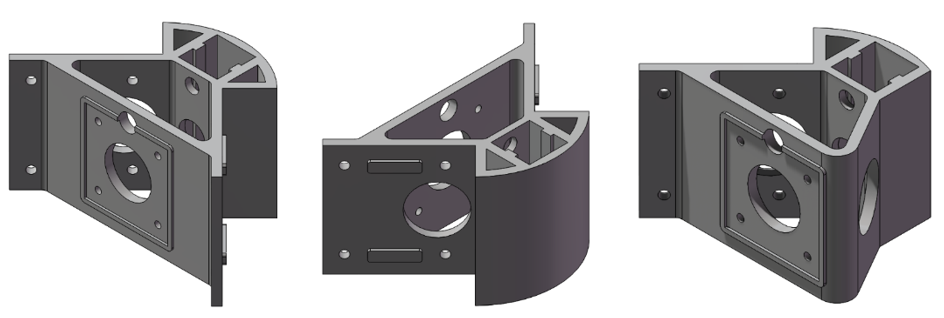
**Figure 28.** 2GT timing belt (left) and 2GT pully (right) (ZYLtech, 2018)

The structure of the design is made up of 2020 aluminum extrusions and injection mold joints, and the joints also work as motor frames. Figure 29 shows an injection mold joints of Kossel printer, the joint provides a 60-degree angle, the extrusion can be fixed through the square hole. The parts are assembled with m4 hex bolts, the advantage of using hex bolts is that hex keys are often design to operate in tiny space.



**Figure 29.** Combined joint and motor frame of Kossel 3D printer

In order to fit the new workspace, the joints are modified and shown in figure 30. Only one side of two of the joints are connected to the extrusion, another one provides 90-degree angle. Since FDM decreases material properties, these parts will only work as the motor frame, besides 6 metal joints are added on the frame.



**Figure 30.** New designed combined joint and motor frame

The most unique designed part of the design is the carriage. The shape of the carriage is designed to fix the belt, which is shown in figure 31. The carriage is fixed on the linear guide, at the top of the guide a limit switch is fixed, and a long wire is needed for the limit switch.



**Figure 31.** The carriage of the modified Kossel 3D printer

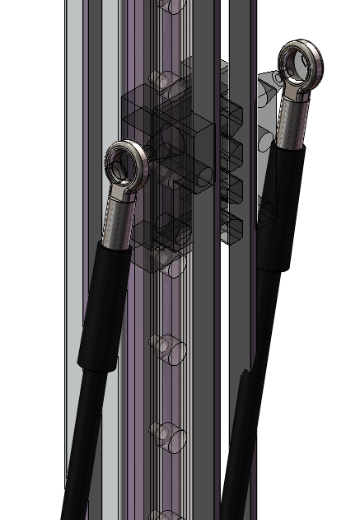
The mechanism chain structure of the origin product is reserved, the chain length is 264mm, because the rods are standard parts. As is shown in figure 32, the terminal of the rod is linked with a ball joint. An M3 bolt and a gasket is used to assemble the mechanism chain which can provide the maximum angle of , and the length of workspace can be calculated with equation 11 which means standard B5 paper can be used as the workspace.

|  |  |
| --- | --- |
|  | (11) |



**Figure 32.** Terminal of the mechanism chain

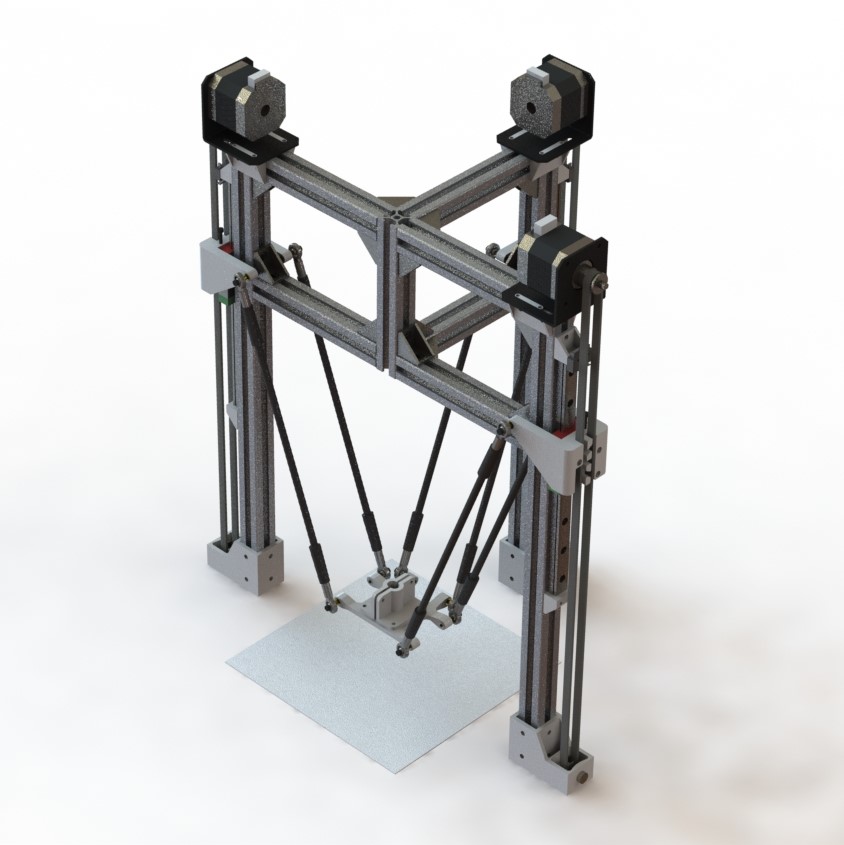
The drawback of the design is obvious too, the belt interference (figure 33). The interference occurs when the end effector is approaching the corner of the workspace.



**Figure 33.** Timing belt interference

### Alternative design

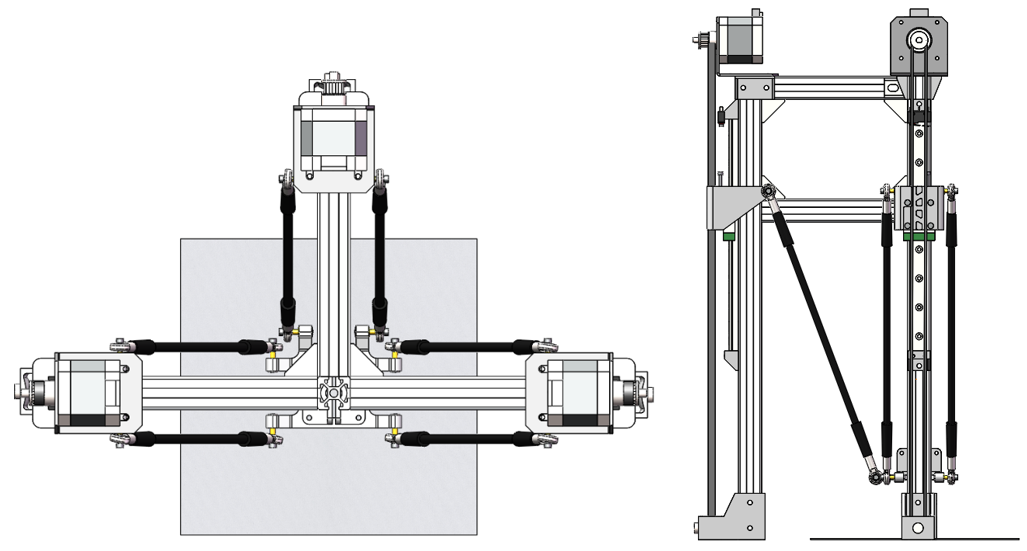
The second design is aimed to solve all the problems raised above and the render graph is shown in figure 34. There are two main features, one is to place the stepper motor at the top of the robot, another is to reallocated the tower structure so that belt interference can be avoid. As considering the demand, the working plane is coincided with the mounting plane of the robot.



**Figure 34.** Render graph of the new designed linear delta robot

Since the mechanism chain length is same as the previous design, the workspace is still 180\*180mm.

381mm\*203mm

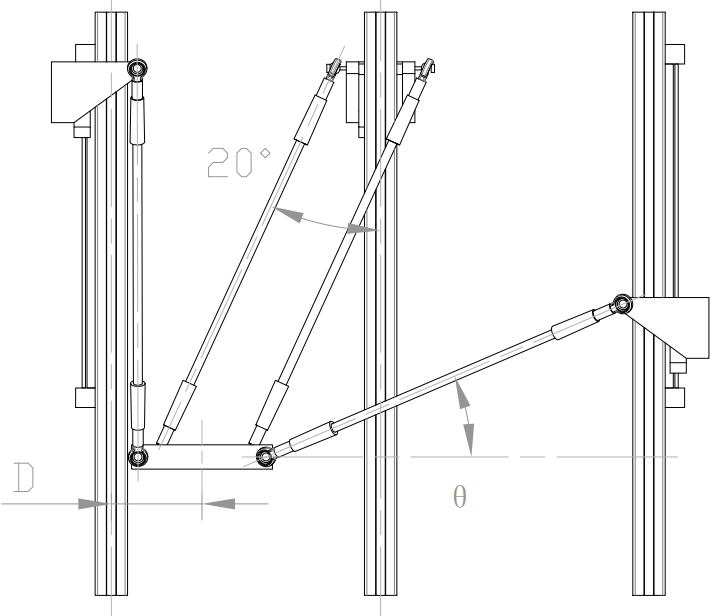


**Figure 35.** Top view (left) and side view (right) of the new designed robot

### Robot framework

The robot framework is constructed with 20\*20mm aluminium extrusions and all the joints are replaced with standard metal 90-degree joints. The transmission system is a timing belt system but the timing belt and carriage joint are placed at both sides of the tower and the motor is placed at the top of each tower. As the result, interference will not occur until the mechanism chain is parallel with the tower.

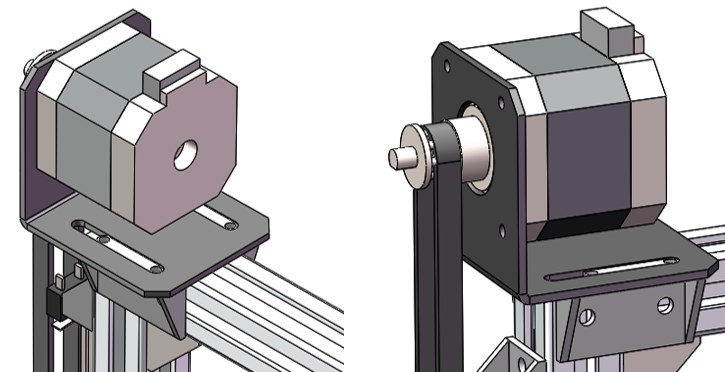
A line graph of the mechanism is shown in figure 36, where D represents the distance between tower center line to end effector center line. In the new design the chain can be vertical when D is at its minimum which means the timing belt interference problem will not appear in this design.



**Figure 36.** Mechanism of the new design

### Motor frame

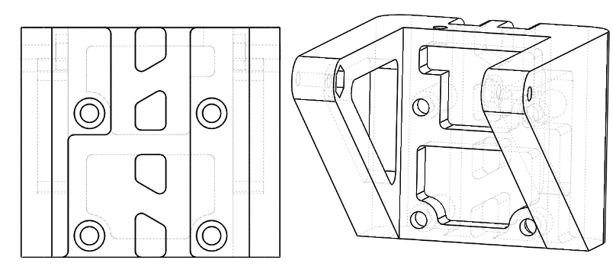
In the origin design, plastic joints are designed to fix the aluminum extrusions as well as the stepper motor. As considering the strength difference and precision between injection molding parts and 3D printed parts, a metal standard motor frame is selected. Besides, two 3D printed supports are designed to fix the motor frame onto the aluminum frame (figure 37)



**Figure 37.** Stepper motor and motor frame

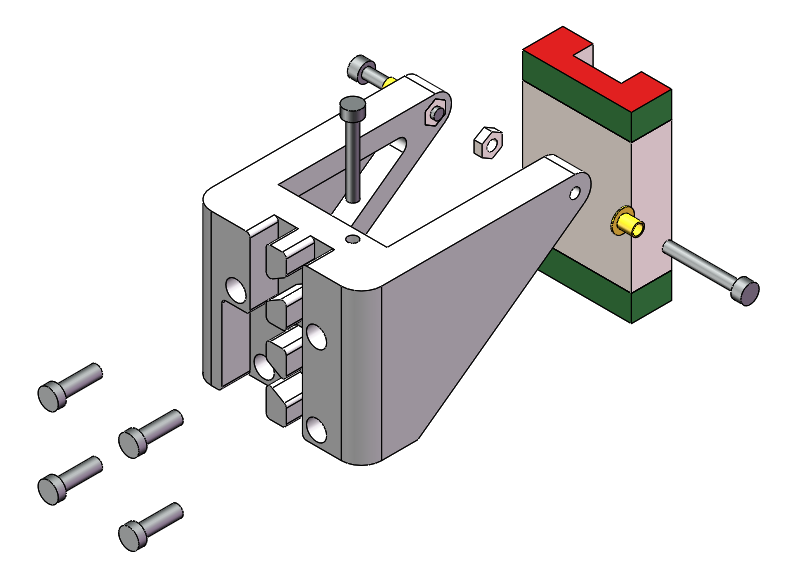
### Carriage

The carriage reserved the features of fixing the timing belt from the current product which is shown in figure 20. The two cantilever holders are the new designed features, which connect the terminals of mechanism chain. The part is shelled to reduce the weight and manufacturing cost, and hexagon holes are designed to improve the convenience of placing the nuts.



**Figure 38.** Line graph of the carriage

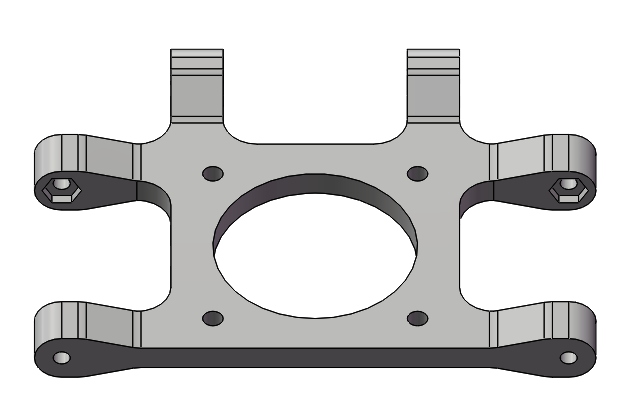
The carriage can be fix on the slider of a linear guider with four M3 bolts (figure 39); the method of connecting with mechanism chain is same as the method explained in section 3.2.1; another M3 blot that passes through the carriage is used to trigger the limit switch when it reaches the reset position.



**Figure 39.** Explore graph of carriage assembly

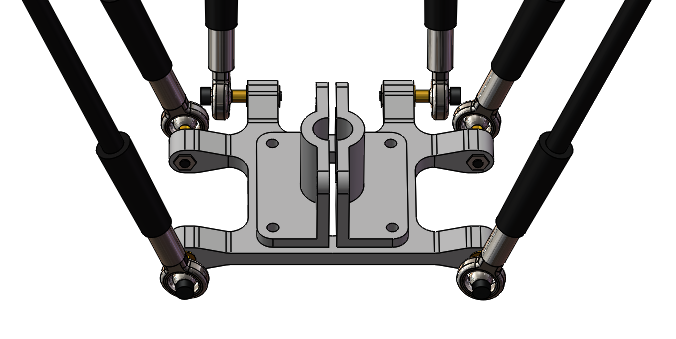
### End effector

The main part of the end effector is shown in figure 40. The features of connecting the mechanism is same as the carriage so that the two rods can be parallel. The minimum thickness of the part is designed to be 3mm, and a big hole is designed to contain tools as well as reducing the weight.



**Figure 40.** The main part of end effector

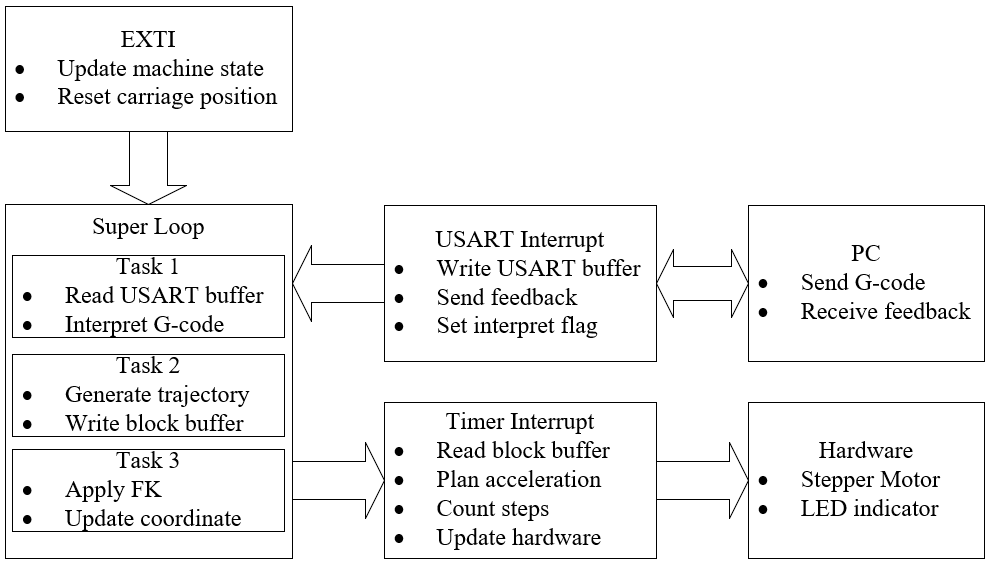
The assembly graph is shown in figure 41, where the two plastic parts that covers the hole is used to fix a pencil.



**Figure 41.** The end effector assembly

## Control method

The control system includes three tasks and several interrupts, since there is no complex priority between the tasks, a typical foreground / background system which a super loop can be applied in the main function rather than using RTOS (real time operation system). The following paragraph will explain the operation process and the function of each module briefly.



**Figure 42.** Flowchart of the control system

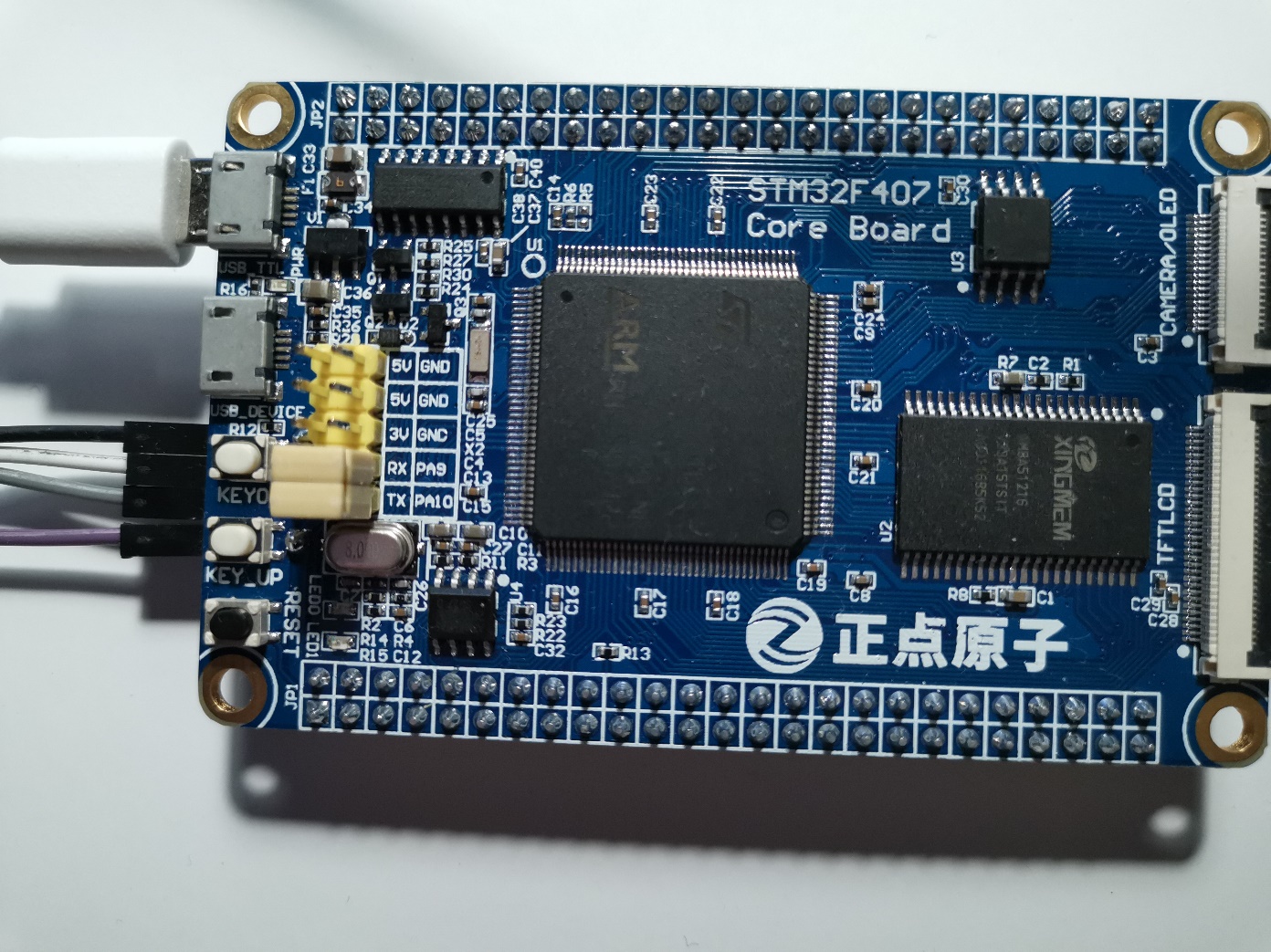
The G-code command is sent from a GUI (graphic user interface) that runs on a PC through the serial port. The GUI is not a part of the design and is quote from XXX, it has the function of sending and receiving characters in the terms of Ascii codes. When the microcontroller receives data bits from the port the USART interrupt function will save these data into a circular buffer and then set the interpret flag.

Interpreting G-code is the first task in the super loop, which will be executed if the interpret flag is set and the details about the interpretation will be explained in section 3.3.2. After interpretation, the second task will be executed to generate a trajectory, in which the path and trapezoidal profile will be generated and inverse kinematics will be applied. The trapezoidal acceleration and deceleration profile will be calculated twice which will be explained in section 3.3.8. The third task is seemed as a feedback task, in which forward kinematics will be applied to update the end effector position.

The timer interrupt that works as a monitor has the second higher priority, the timer interrupt function is executed periodicily begin with reading a block from the block buffer, then plan the acceleration and deceleration steps. The external interrupt has the highest priority which can be triggered by limit switch and stop and reset button.

### Serial communication

In this section, method of using serial communication to receive and send information on STM32 will be explained. However, the algorithm in programming will not be focused, because the serial lib provided by Alientek has already optimized and meet the demand. Fortunately, a protocol chip is embedded on the board, which convert RS232 to micro USB. So, the only job needs to do is to connect TX and RX terminal with two GPIOs which is shown in figure 43.



**Figure 43.** Serial port of STM32F4 board

The following code is the USART interrupt service routines which will be executed automatically as soon as serial port receives data. The first step is to check and reset the interrupt pending bit; then the character that received from the serial port will be saved in a temp variable “res”; when the value of the received variable is the ascii code of carriage return, the interpret flag will be set and the interpretation task will be executed in the main routine when the interrupt is executed.

void **USART1\_IRQHandler**(void)

{

    uint8\_t res;

    if (**USART\_GetITStatus**(USART1, USART\_IT\_RXNE)!=RESET)

    {

**USART\_ClearITPendingBit**(USART1, USART\_IT\_RXNE);

        res = **USART\_ReceiveData**(USART1);

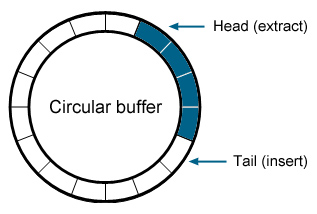
**Uart\_Buff\_Write**(&uart\_buff,res);

        if (res==13)    machine.interpret\_flag = SET;

    }

}

The USART buffer is designed as a circular buffer, which is shown in figure 3.4.2. Circular buffer is a type of FIFO (first in first out) data structure, which can be presented as a array with a constant length and two pointers. The head pointer points to the value that will be read first, and the tail pointer points to the position that a new value will be wrote. When a value is read from the buffer, the head pointer points to the next address which means the read value will not be cleared but will be covered automatically when the tail pointers pointes that position.



**Figure 44.** Circular buffer

The advantage of using the circular buffer is that its memory is continuous, which increases the reading speed. Moreover, a structure is designed to put the array, pointers and buffer length together which is shown in the following code.

typedef struct

{

    uint16\_t    head;

    uint16\_t    tail;

    uint16\_t    length;

    uint8\_t     content[100];

}**uart\_buff\_t**;

### 3.3.2. G-code interpreter

The G-code interpreter is used to interpreter the characters that stored in the serial buffer to values to specific command. The following code is C structure of the command and it is the output of the interpreter, where xyz array is the coordinate of target position and command type includes home command, linear motion command, circular motion command and dwell command. The circular motion is clockwise if the value of radius is positive.

typedef struct **\_command**

{

    float xyz[3];

    float feedrate;

float radius;

float dwell;

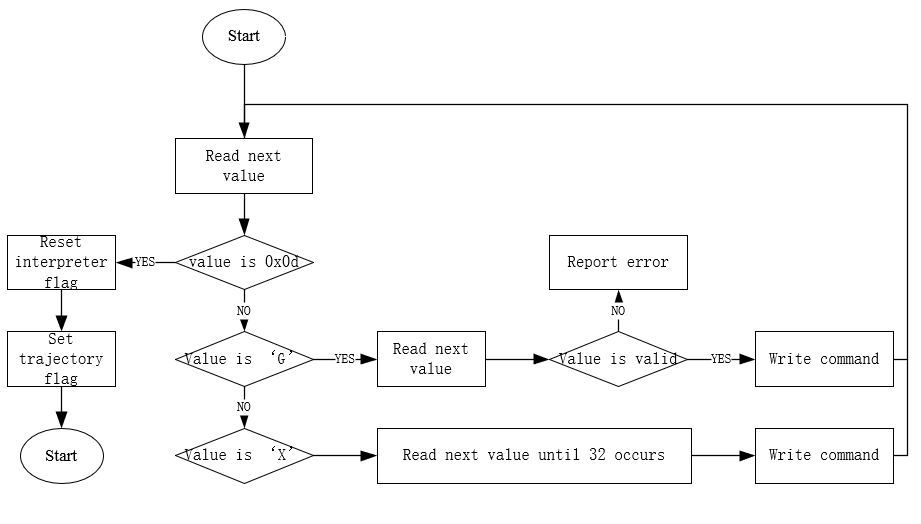
    uint8\_t type;

}**command\_t**;

enum **Motion\_Type**

{home\_t = 0, linear\_t, arc\_t, dwell\_t};

The flowchart of the interpreter is shown in figure 45. The method is to get the values after a key character that is listed in table 6and convert them from integers to floats. For example, if the key character is ‘G’, then all the characters after ‘G’ will be considered as a part of the result until the ascii code of spacebar is read.



**Figure 45.** Flow chart of the G-code interpreter

|  |  |  |
| --- | --- | --- |
| Key character | Represents | Variable type |
| ‘G’ | G command type | char |
| ‘X’ | X coordinate | float |
| ‘Y’ | Y coordinate | float |
| ‘Z’ | Z coordinate | float |
| ‘P’ | Dwell in milliseconds | float |
| ‘R’ | Radius | float |
| ‘F’ | Feedrate | float |

**Table 6.** G-code key characters

### 3.3.3. Inverse kinematics

The method of solving inverse kinematics problem begins with creating a coordinate system on the base frame (figure 46) and another one on the end effector (figure 47). Assume the position vector of point in base frame is:

|  |  |
| --- | --- |
|  | (12) |

As is shown in figure XXX, the three points represents the position that connects a mechanism chain, and the coordinate of the point Pi in the end effector frame E can be described by the vector , which can be expressed as

|  |  |
| --- | --- |
|  | (13) |

Where

Then, the coordinate of the point Pi in the base frame B can be described by the vector from figure XXX, which can be expressed as

|  |  |
| --- | --- |
|  | (14) |

By using the same method, the coordinates of carriages in the base frame can be expressed as

|  |  |
| --- | --- |
|  | (15) |

Since the position of Pi and Ci in base frame is defined, and the distance between these two points is the length of mechanism chain L, so the inverse kinematics can be solved:

|  |  |
| --- | --- |
|  | (16) |

That is,

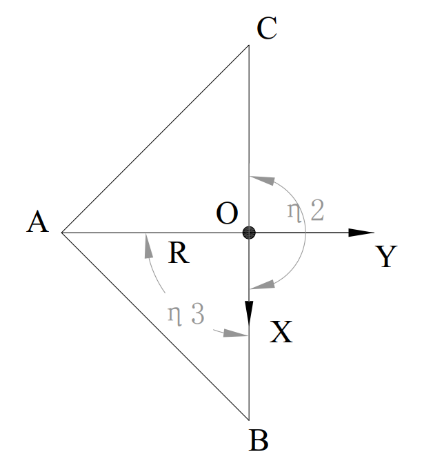
|  |  |
| --- | --- |
|  | (17) |

where, , , are the inputs, the position of the carriages. As considering the geometry of robot parts which has been explain in section 2.2, the equation can be expressed as:

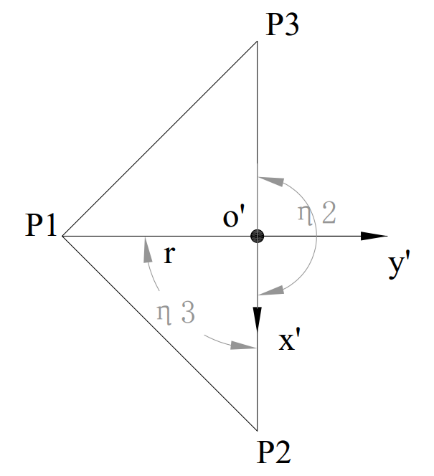
|  |  |
| --- | --- |
|  | (18) |

where, , , H is the height of end effector, C is the carriage offset. Meanwhile, the carriage is always above the end effector so rearranging equation XXX, the expression of input zi can be described as:

|  |  |
| --- | --- |
|  | (19) |



**Figure 46.** Top view of the base frame



**Figure 47.** Top view of the end effector frame

### 3.3.4. Jacobian matrix

As is described in section, Jacobian matrix maps the velocity of end effector and the velocity of carriages. Rearranging equation XXX, the expression of Jacobian matrix is:

|  |  |
| --- | --- |
|  | (20) |

where is the vector of end effector

|  |  |
| --- | --- |
|  | (21) |

and is the vector of carriages

|  |  |
| --- | --- |
|  | (22) |

Equations XXX can be differentiated with respect to time to obtain the velocity equations, which leads to

|  |  |
| --- | --- |
|  | (23) |

Then, the Jacobian matrix can be defined:

|  |  |
| --- | --- |
|  | (24) |

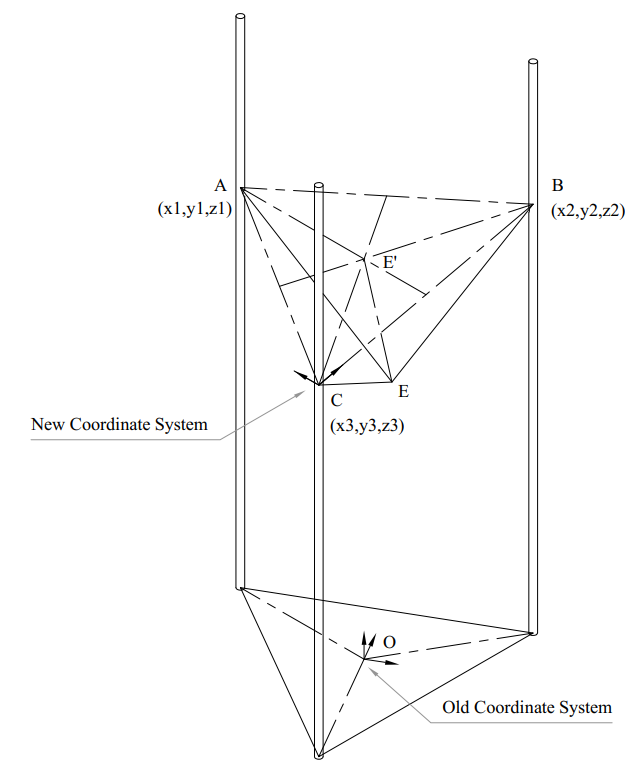
### 3.3.5. Forward kinematics

Solving forward kinematics problem is to find the end effector coordinates when the carriages positions are defined. Generally, there are two methods, one is to rearrange the inverse kinematics equation and solve ternary quadratic equation (25). The ternary quadratic equation has the same value range as the inverse kinematics equation.

|  |  |
| --- | --- |
|  | (25) |

The second method is trilateration which has been introduced in section 2.7.3, and it has been applied on Marlin firmware. As considering the geometry meaning of equation 18, then it is clear that the equation describes three spheres, and their center point can be presented as .

These points are not the position of the carriage, but their distance to the end effector center is the length of mechanism chain. As is shown in figure 48, the three points are presented as , E represents the end effector, and E’ is the projection of point E on plane . The problem is to find the coordinate of point E or vector in the old coordinate system, when the coordinates of point in the old coordinate system is given, the distances from point E to the carriages are known.



**Figure 48.** Forward kinematics model of a linear delta robot

The method is to establish a new coordinate system in the plane of the three carriage points., which has its origin at tower C, with tower B on the X’ axis. Tower C is in the X’-Y’ plane with a Z component of zero. Then apply trilateration method to find which is the vector of point E according to the new coordinate system. Finally, vector can be determined by applying the following calculation

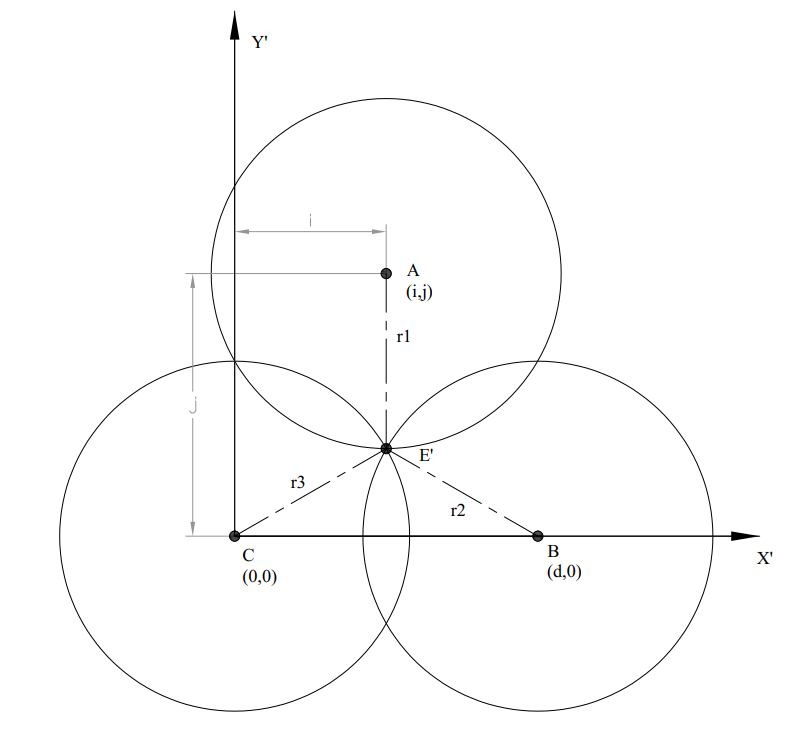
|  |  |
| --- | --- |
|  | (26) |

The new coordinate system is shown in figure 49. Assume point E’ is the focus of three circles, then radius of the circles can be presented as:

|  |  |
| --- | --- |
|  | (27) |
|  | (28) |
|  | (29) |

As is mentioned, , and vector in the new coordinate system can be determined:

|  |  |
| --- | --- |
|  | (30) |
|  | (31) |
|  | (32) |



**Figure 49.** New system built in forward kinematics model

Then, the problem becomes to determine the value of , , and vector . By creating a vector in old coordinates along x‘ axis of new coordinate.

|  |  |
| --- | --- |
|  | (33) |

D can be determined by getting the reciprocal of Magnitude of vector:

|  |  |
| --- | --- |
|  | (34) |

Create unit vector by multiplying by the inverse of the magnitude.

|  |  |
| --- | --- |
|  | (35) |

Get the vector from the origin of the new system to the column A.

|  |  |
| --- | --- |
|  | (36) |

Use the dot product to find the component of this vector on the X axis.

|  |  |
| --- | --- |
|  | (37) |

Create a vector along the x axis that represents the x component.

|  |  |
| --- | --- |
|  | (38) |

Subtract the X component from the original vector leaving only Y.

|  |  |
| --- | --- |
|  | (39) |
|  | (40) |
|  | (41) |

Use cross product to find z vector in the new coordinate system.

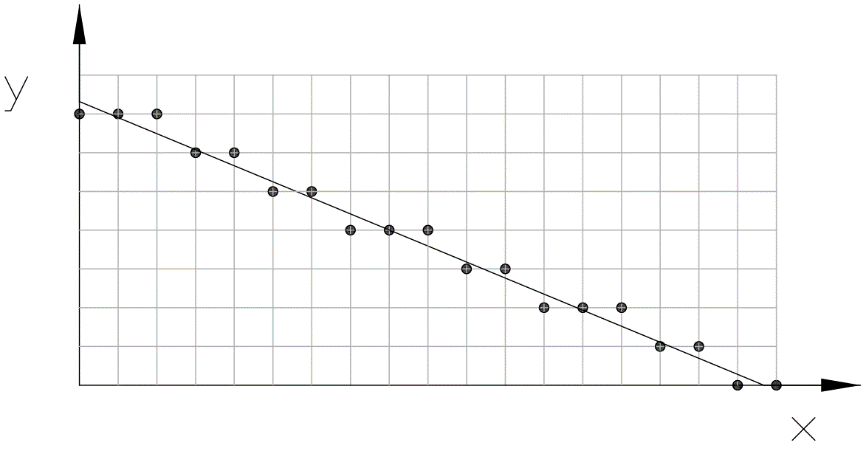
|  |  |
| --- | --- |
|  | (42) |

Hence, coordinate of point E’ in the x’-y’ coordinate system can be determined

|  |  |
| --- | --- |
|  | (43) |
|  | (44) |
|  | (45) |
|  | (46) |

### 3.3.6. Linear trajectory

The idea of generating linear path is using the Bresenham’s line algorithm which is shown in figure 50. According to the method that is mentioned in 2.6.2, the points can be considered by comparing their distance between the ideal path, however modulus operation in C language gives another choice.



**Figure 50.** CAD drawing of Bresenham’s line algorithm

The modulus operation can be used to find the remainder of two integers. The judge statement can be written as:

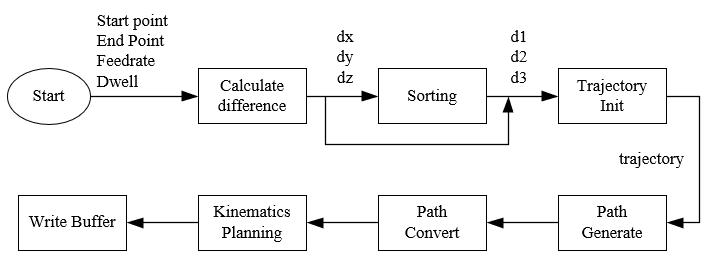
|  |  |
| --- | --- |
|  | (47) |

Where C is the condition, is the resolution which is the length of the matrix, is the y offset between initial point and target point, is the x offset between initial point and target point, is the sequence number of the points. The sub movement

As the path is same with the one shown in the picture, the y offset of next point should be minus the resolution when the remainder is smaller than half resolution. Otherwise, the y offset will not change. However, there is one drawback of using this method which is the condition of applying this algorithm is that x offset is larger than the y offset, as a result, a sorting function must be applied before generating the path.

The input of the linear motion function includes a start point, a target point, the federate and dwell time before operation. Figure 51 is a flow chart of the function.

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXx

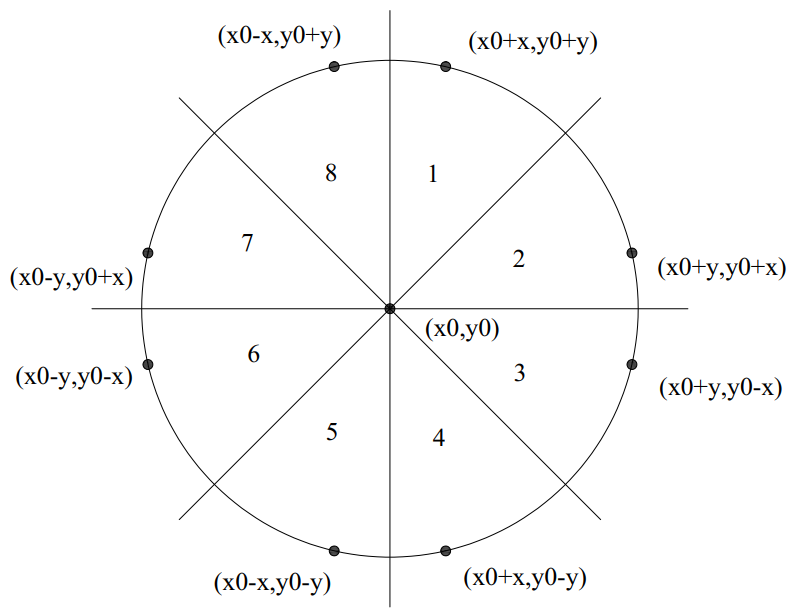


**Figure 51.** Flow chart of linear trajectory planning algorithm

### 3.3.7. Arc trajectory

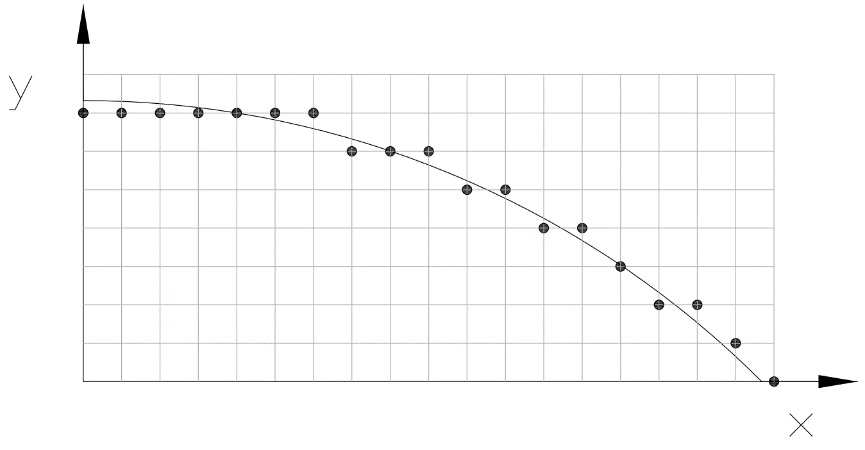
The method of generating arc path that is used on GRBL firmware is to divide the arc into a set of linear paths, and then generate the path with the linear path function. Such method has two drawbacks, first arc with small radius (lower than 5mm) can not be applied; second feed rate is low.

Generally, there are three methods to handle with arc motions, the Pythagorean, Polarize and Bresenham’s circle algorithm. But at the beginning, it is necessary to divide a circle into 8 sections which is shown in figure 52, because once a point and the pivot is defined, another seven points can be found with a simple conversion. Moreover, the drawback that met in Bresenham’s line algorithm can be also solved by the way, because the derivative of the path is always smaller than 1 in the first sector.



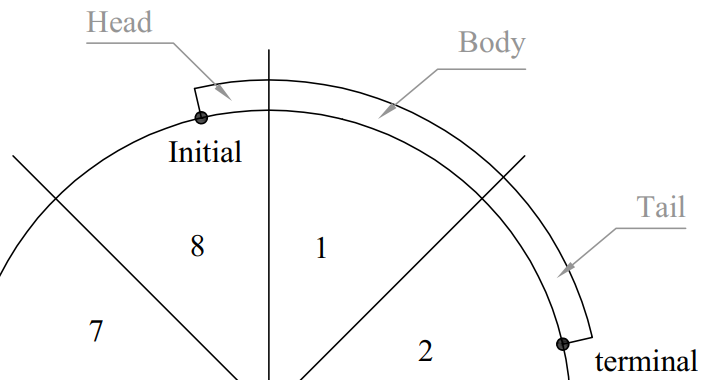
**Figure 52.** Relationships between the points in a circle

However, the situation that met in realistic is much complex than doing the sector conversions. As the result, the path can be classified into three cases as considering the position of the two terminal points and the arc length.



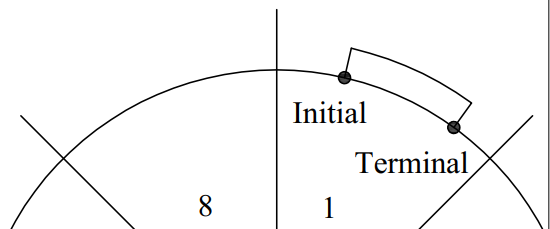
**Figure 53.** Bresenham’s circle algorithm on circular path

In the first case, the path length is longer than the length of a full sector path. In such situation, at least on full sector must be covered by the path, which means the path consists a head, a body and a tail. So, the first step is to generate the full path of sector 1 and then fill in the points in body path with sector conversion. The second step is to generate the head and the tail by cutting a segment from the full sector path. The result includes three paths, but it is not necessary to do the stitch because they are continuous.



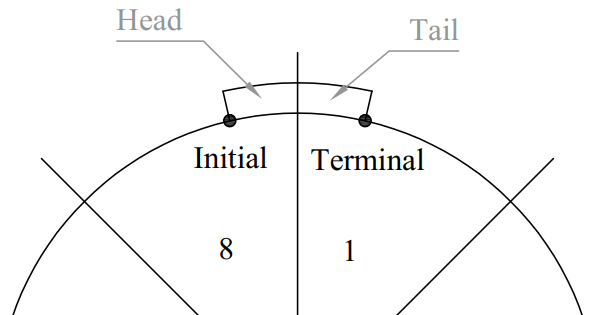
**Figure 54.** Circular path case 1

In the second case, the arc length is smaller than the length of a full sector path and both terminal points are in the same sector. In this case generating full path of a sector is inefficient or even unpractical, for example, a path with small length but large radius may waste a lot of time. Therefore, in such case the path should be generated directly.



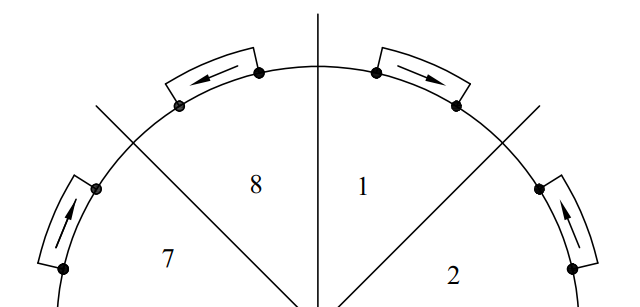
**Figure 55.** Circular path in case 2

In the third case, the arc length is smaller than the length of a full sector path, but the two terminal points are located in two adjacent sectors. It is inefficient to generate both parts individually using the method that is used in case 2, because0



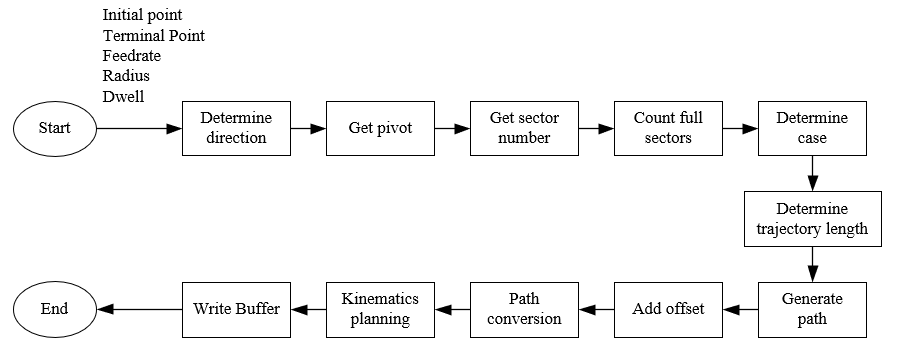
**Figure 56.** Circular path in case 3

Another important method that haven’t mentioned yet is sector conversion. As is shown in figure 57, the sequence of the points of a path is opposite between any two adjacent sectors. As a result, it is necessary to determine the sector number of the initial point and terminal point especially in case 1.



**Figure 57.** Path directions in different sectors

The flow chart of generating circular trajectory is shown in figure 58.



**Figure 58.** Flow chart of generating circular trajectory

### 3.3.8. Acceleration planner algorithm

The planner algorithm is used to plan the acceleration and deceleration detail of each blocks, and the algorithm will be applied in the main loop. Before explaining the algorithm, it is necessary to introduce to the input, which is a trajectory that is presented as a triple-pointers array:

    float traj[len\_traj][3][2];

The first pointer points to the sequence number of the block, while the second one points to the motor ID and the third one points to the carriage position and its velocity. The output of the algorithm is a block which is shown below.

typedef struct

{

*//generate by planner*

    uint8\_t     dir[3];

    uint32\_t    entry\_freq[3];

    uint32\_t    maximum\_freq[3];

    uint32\_t    leave\_freq[3];

    uint32\_t    accelerate\_until;

    uint32\_t    decelerate\_after;

    uint32\_t    accelerate\_rate[3];

    uint32\_t    decelerate\_rate[3];

*//update in ISR*

    uint32\_t    step[3];

    uint32\_t    step\_dwell;

    uint8\_t     flag;

}**block\_t**;

The block calculation will be executed twice, and in the first routine the direction and step amount can be easily determined by applying equation XXX, and the sign of the result represent the carriage direction.

|  |  |
| --- | --- |
|  | (49) |

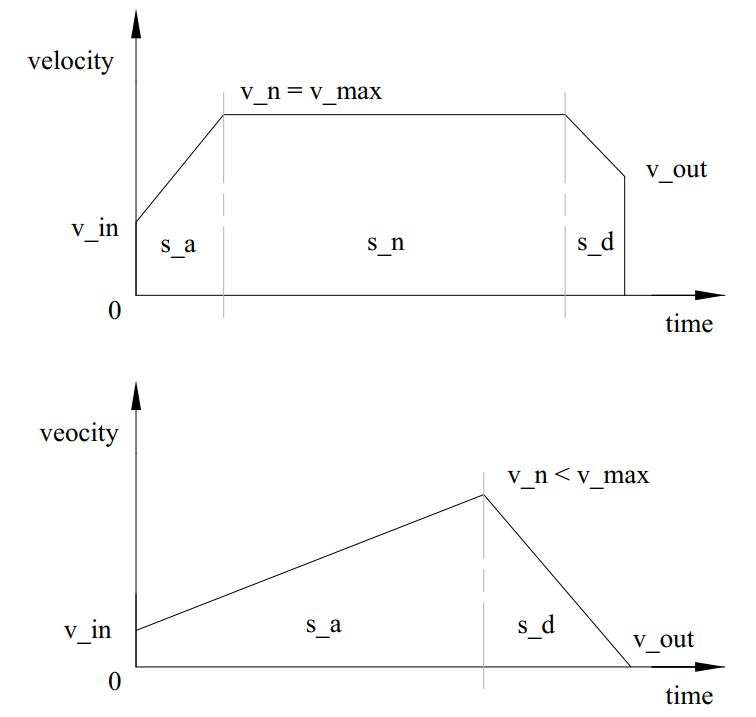
Where, is the current carriage position [mm], is the last carriage position [mm], is the resolution [mm]

Then, the velocity that calculated with Jacobian matrix is defined as the maximum velocity in millimetre per second. While, the velocity can be convert to the stepper frequency easily with equation 50.

|  |  |
| --- | --- |
|  | (50) |

Where, is the tooth length, is the amount of teeth on the pully, is angle of a unit step

The entry frequency equals to the leave frequency of the last block, and the leave frequency of a new block is set to 0, because all the motors have to stop after executing the last block. There is no adopt that the motors should accelerate as fast as possible, but all the motors should both accelerate and decelerate at the same time, which means the proportion between the acceleration of each motor should be the same as their frequency difference proportion. Moreover, the acceleration of the motor that has the largest frequency difference should be the maximum acceleration that the motor allowed. When the accelerations and decelerations are determined, the steps that needed can be also worked out.



**Figure 59.** Triangular (left) and trapezoidal (right) speed profile

Assume the motion cure meets the triangular condition, then the total distance is the sum of the acceleration distance and deceleration distance which can be presented:

|  |  |
| --- | --- |
|  | (51) |

And the velocity equation can be presented:

|  |  |
| --- | --- |
|  | (52) |

Rearranging equation 51 and equation 52, the acceleration time can be calculated:

|  |  |
| --- | --- |
|  | (53) |
|  | (54) |
|  | (55) |

Then, it is straight forward to calculate the peak velocity:

|  |  |
| --- | --- |
|  | (56) |

If the peak velocity is larger than the maximum velocity, then the motion profile is trapezoidal, and the acceleration time should be calculated again:

|  |  |
| --- | --- |
|  | (57) |

And deceleration time:

|  |  |
| --- | --- |
|  | (58) |

The distance travelled during acceleration and deceleration can be determined:

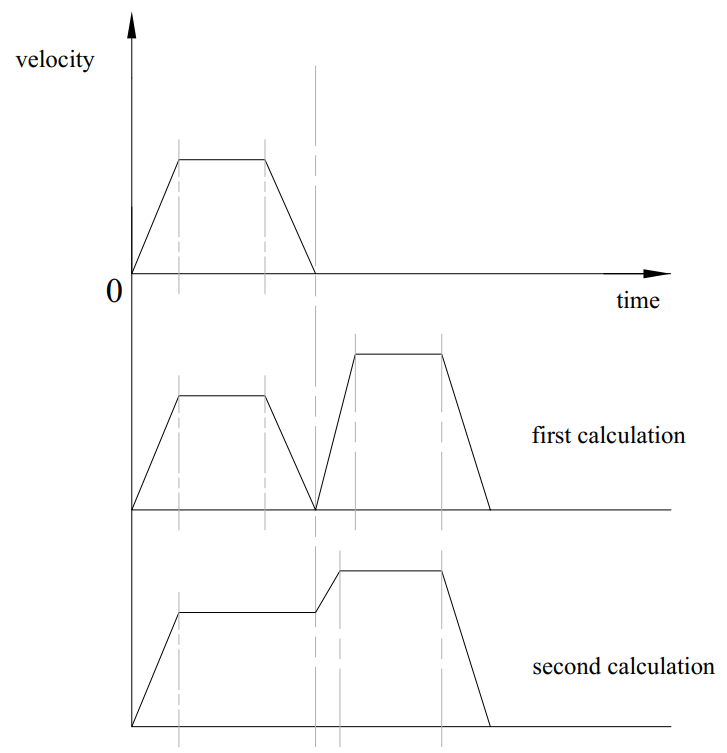
|  |  |
| --- | --- |
|  | (59) |
|  | (60) |
|  | (61) |

Hence, the distance travelled in all stages are defined. However, if the peak velocity is smaller than the maximum velocity, which means the motion profile is triangular and the acceleration time do not need to be calculated again. The distance can be calculated by repeating equation 59 and equation 61.

### 3.3.9. Speed connection

As is mentioned in section 3.3.8, the acceleration planner algorithm will be applied twice, and the reason of that is to connect the speed between each block. As is shown in figure 60, the first block that is added to the buffer describes a trapezoidal speed profile that accelerate from 0 and decelerate to 0. When another block is added, the entry speed of the new block should meet the leave speed of the last block, and the leave speed of a new block is always set to zero, because the motor must stop when the block buffer is empty.

Therefore, if the acceleration is only calculated once, the motor has to stop as soon as a block is executed which decreases the execution speed. During the second calculation, the maximum speed of each block will be compared, and the connection part can be either acceleration or deceleration.



**Figure 60.** Connect the speed when a new block is added

### 3.3.10. Stepper control algorithm

The purpose of the stepper motor control algorithm is to update the motor parameters in real time, so that they can keep the preset speed and acceleration. As a result, the algorithm will be applied in a timer interrupt function and that means the algorithm cannot include complex calculations or float operations.

According to the method introduced in XXX, the principle of generating pulse signal is to set and reset an output bit and then update the timer delay. However, this algorithm has a drawback which is another algorithm needed to be designed to manage the delays when the amount of motors is increasing. Besides, there is another way which is using the PWM generating function.

The PWM function is based on timers, and the timers selected are called general-purpose timers which consist of 16-bit auto-reload counter driven by a programmable prescaler. The timers are completely independent, and do not share any resources, so that the efficiency of executing the main loop can be improved.

The function of calculating the timer is:

|  |  |
| --- | --- |
|  |  |

Where is the input frequency 84MHZ, is the overflow time, is the prescaler and is auto reload register .

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It

is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx\_PSC

register). It can be changed on the fly as this control register is buffered. The new prescaler

ratio is taken into account at the next update event.

Figure 54 shows how an acceleration is applied. The top of the figure is a v-t graph which represents how the motor or carriage accelerate from the entry speed to operation speed. The entry speed can be easily got from the pre scalar register of the corresponding stepper motor.

XXXXXXXXXXXXXXXXXXXX



**Figure 61.** Principle of motor driver algorithm

# Results

In this chapter, the results of this project will be presented. First, a set of pictures of the prototype will be displayed; then the acceleration curves that plotted with STM Studio will be shown to find out how the motors respond to a command, at the same time, the IO state will be plotted as the PWM wave; after that, the result of kinematics simulation with Matlab will be presented.

## Mechanism of the robot

The overall design is shown in figure 4.1.1. The robot weights 5 kg

Figure 4.1.1 – The prototype

The carriages, end effector and other small parts are made up of UV Curable Resin.

Figure 4.1.2 – 3D printed parts

The end effector is now fixed with a pencil, which can be used to record the actual path. The distance between the peak and end effector center is 10 mm, and the end effector assembly wights 100g, of which the influence of weight can be ignored as comparing the motor inertia.

Figure 4.1.3 – end effector with a pencil

The circuit is built on a breadboard, which consists of three A4988 stepper motor drivers. All the chips are covered with radiator

Figure 4.1.4 – Circuit of the prototype

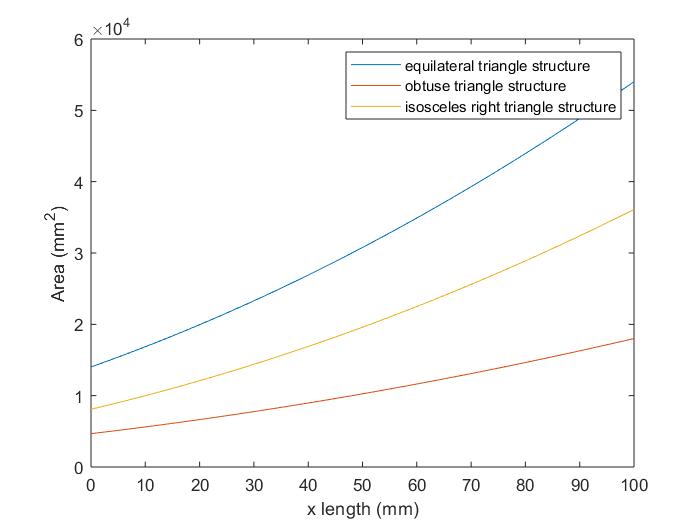
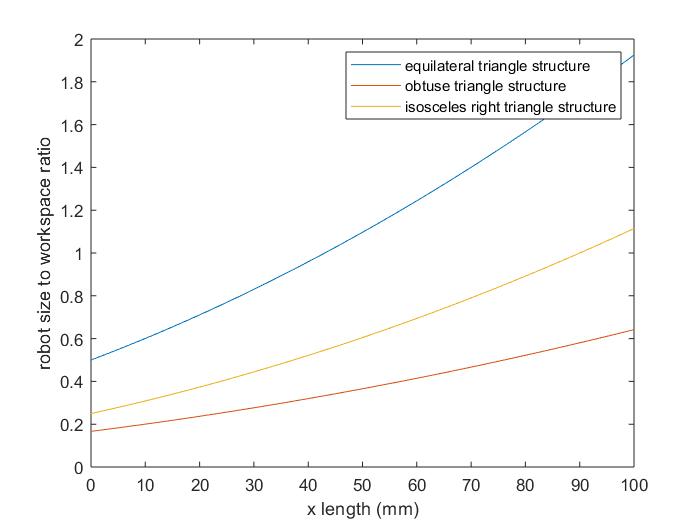
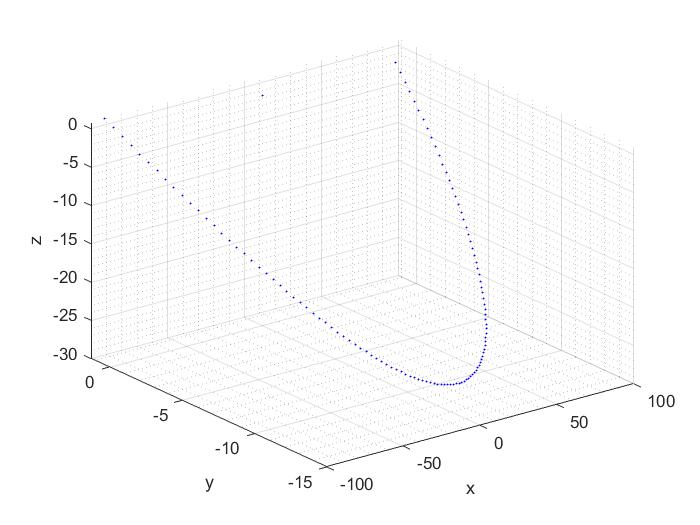


Figure 62. Comparation



## Kinematics simulation

In delta robots, the tower motion is not linear to the end effector motion. For example, assume the end effector is moving from (-90,0,0) to (90,0,0), and carriages moves in a steady velocity. The result calculated with forward kinematics is show in figure XXX, where the maximum offset is XXX. As a result, it is necessary to divide each linear segment into lots of very small subsegments (blocks).



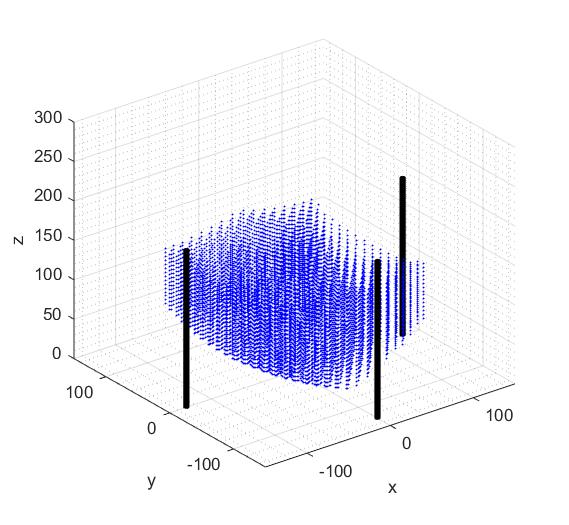
**Figure 63.** End effector path when the carriage is applied a linear motion

The key word “block” is firstly used in GRBL firmware to describe a small segment of motion in which the motors keep a steady speed. In other words, a g-code command can be divided into a lot of blocks, of which the amount is determined by the motor resolution. The following code shown a C structure that is used this design as a block.

A block includes the direction, speed of each motor, the step count and the dwell step count; the motors start to work when the dwell step count is zero and stop when the step count is zero. The flag is used to indicate whether the it is the last block of a g-code command. The blocks are stored in a circular buffer which has the same principle with the USART buffer, because the reading speed could limit the motor update frequency.

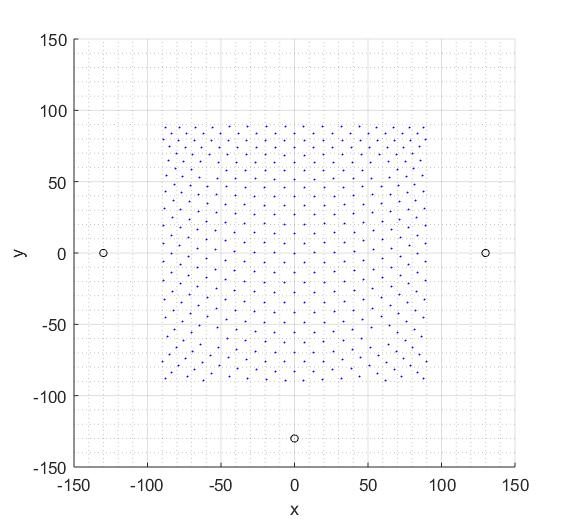
However, such method is not suitable for numerical methods. Here is an experiment to test execution time of solving ternary quadratic equations on Matlab. The step length of each linear actuator is set to 10mm and the motion limit is from 250mm above ground to 500mm, as is shown in figure XXX, 27001 points are plotted and the time consumption is 4.75 hours. It is straight forward to work out that it takes 0.63 second on each point on the path.

All points on matlab, find improvement

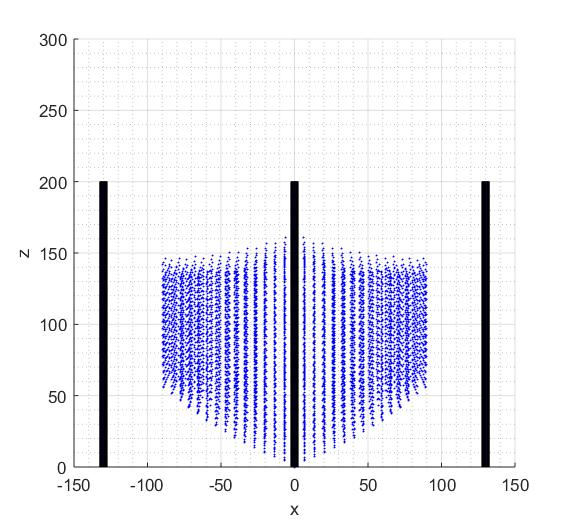


**Figure 64.** Workspace that simulated with Matlab

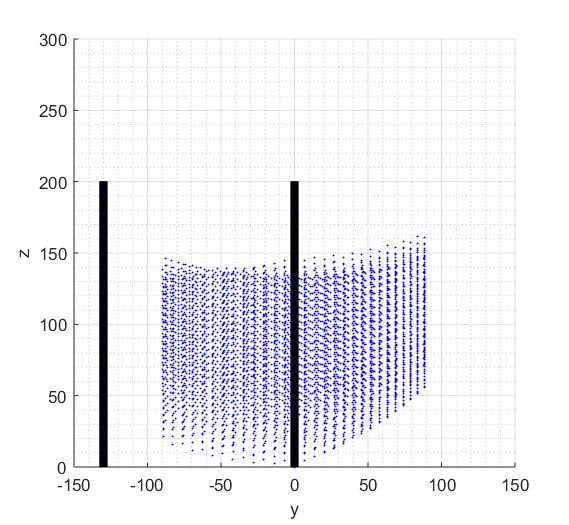
Figure XXX is the x-y sight of the workspace. The result shows that plotted points can be more suitable to a cartesian coordinate system if the cartesian coordinate system is rotated 45 degree around z axis in either directions. Besides the distance between each points is much wider than those close to the edge. As a result, errors must exist if the cartesian coordinate system is applied directly onto the path generator. On the other hand, it is necessary to add filter to minimize the average error.



**Figure 65.** Top view of the simulated workspace



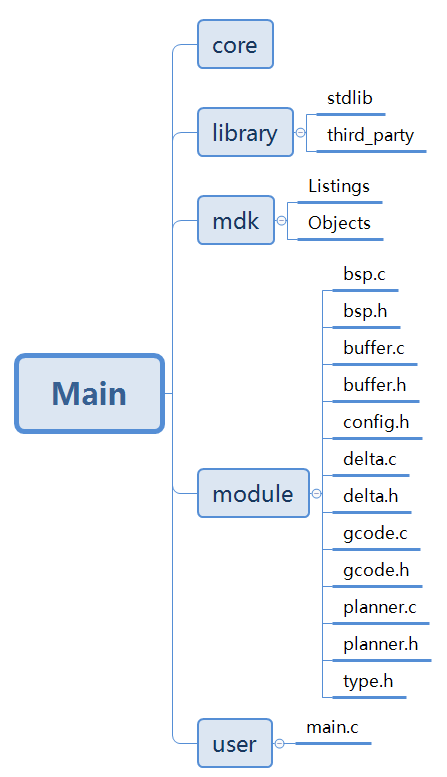
**Figure 66.** Front view of the simulated workspace



**Figure 67.** Side view of the simulated workspace

## Program structure

The logic diagram of file and file folder structure is shown in figure XXX. The core folder consists startup files of the microcontroller which is written by the chip manufacturer; the library folder includes STM32 standard library and third party library; the mdk folder is prepared for Keil uVision5 MDK software which generates a lot of temp file in the listings folder and the target file in objects folder; the module folder consists of functions for multi-purpose, and the user folder consists the main file.



**Figure 68.** File and folder structure of the program

There are five C program files and seven head files in the module. The bsp file is the board support package which includes all the driver function that written with STM32 standard library; the buffer file includes all the data structure related functions; the config head file includes configuration information such as robot size and motor maximum speed; the gcode file only includes two functions: inverse kinematics and forward kinematics; the planner file includes all the trajectory planning and speed profile planning functions; the type head file includes short-cut macros and C structures.

The delta function, buffer function, board support package and the timer interrupt routine has been debugged, and the codes are shown in the appendix

# Discussion

In this chapter, the results and findings will be discussed and evaluated. The evaluation will be carried out around the following themes: whether the design can be improved to mass production; advantage and disadvantage of the technical that is used on this project as comparing with current commercial products; feasible method to improve the robot performance.

## Robot structure

The previous paragraph has approved that building a delta robot with isosceles right triangle shaped end effector is feasible, and this section will discuss and evaluate the mechanism and workspace analysis method

### Workspace analysis method

The method used in defining workspace in previous studies is to first solve the inverse kinematics equations and convert it to the forward kinematics equation, after that the workspace can be found by with Matlab simulation. Such process has also been used in this project as the approvement.

One of the achievements of this project is to define the workspace begins with considering the geometry limits, and the model built in section 3.1 are much straight forward to understand the

Previous studies had already explained the general processes of analyzing the workspace of a linear delta robot, which is first define the forward kinematics equation and then add limits on it. However, in this section the analyzation begins with the joint limits, which is much closer to engineering problems.

The biggest different between

//等腰直角完全可行，打印温度和气味

### Structure optimizes

The biggest difference between a prototype and production model is that the previous one is used to proof the principal and theories work, which means inaesthetic can be relatively ignored.

The method of executing G28 (home) command in Marlin is to drive all the stepper motors in one direction until the switches are triggered.

As considering the function of linear delta robots,

As considering the work of 3D printings, a shell is needed to maintain the

### Assembly

The volume of production depends on the speed of assembly, and the quality is determined by the assembly process. The biggest problem during assembly is to find the position of the limit switch. The position of the switches determines the reset position of each end effector, and the reset position of the end effector is also calculated with forward kinematics equations. As the result, the carriage reset position might be different between each product which means the configuration file should be adjust after measurement. As consider the accuracy of current home 3D printers, in addition to increase the assembly accuracy, using sensors and specific algorithm is another choice.

## Hardware evaluation

This section is to discuss the hardware chosen in this project, to find out whether they are practical

### 8-bit processor and 32-bit processor

The micro controller chosen is STM32F407, which is a 32-bit CPU with 100 IOs, 15 EXTI (external interrupt) lines, 6 UART (Universal Asynchronous Receiver/Transmitter) lines and 10 timers. While, many home 3d printer models are equipped with ATMega2560 which is a 8-bit AVR micro controller. The advantage of 32-bit processor is that it has a wide data pipe, and provides more addresses than the 8 bit one. For example, a single integer occupies 4 bits and have a range of in a 32-bit CPU, but only occupies 2 bits and have a range of in an 8-bit CPU. As a result, programming for an 8-bit CPU should always check the range of each variable in order to prevent overflow. Another reason is that the purpose of this project is to build the prototype which needs hardware resources for innovations and tests, and ATMega2560 is more suitable for mass production models.

### Stepper motor

Although a lot of researches have been done to determine the acceleration, deceleration and maximum speed, it seems that using empirical method based on current product is a better choice. The reason is that it is hard to obtain the complete parameters when purchasing reasonable prized motors, as the result, a lot of experiment need to be carried out to define the acceleration in theoretical method.

## Program evaluation

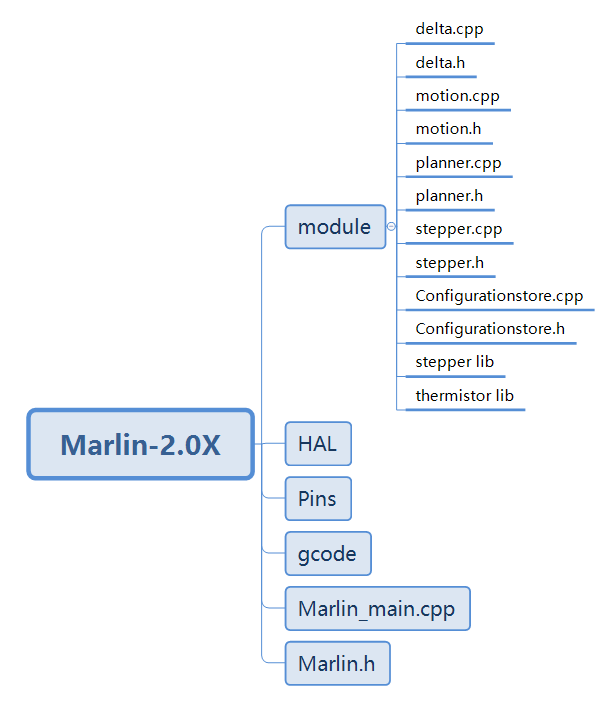
This section is to discuss the

### File structure

File classification is a big problem during programming, because an unreasonable classification may waste a lot of time in updating include path. Besides, a good naming convention of each program file may increase the efficiency on searching functions. The structure of files and folders has been explained in figure XXX, but in this section is necessary to evaluate it with the Marlin firmware.

The combination of

The structure of files and folders of Marlin firmware 2.0X version is shown in figure XXX.



**Figure 69.** File and folder structure of Marlin firmware

### Library comparison

The control program of this project is written with STM32 standard peripheral (SP) library. As comparing with hardware abstraction layer (HAL) library and low-layer (LL) library, SP is the earliest library that is built, the benefit is that it is steady and bug free, meanwhile a lot of sample code and tutorials can be found in open source communities. However, the chip manufacturer has stopped updating SP library, which means fewer commercial products will be developed with this library.

HAL library is replacing the role of SP library, one of the advantages is that it works with STMcube software which converts complex initialization into graphic interface. For example, those who program with SP library have to check the chip booklet before they can initialize the IOs, but STMcube software simplified this process into clicking on graphic interface.

The Arduino library is also needed to be discussed. The Marlin firmware that is researched are usually compiled with Arduino IDLE.

The Marlin shows large benefits of using object oriental programming and pointer operation. Data operation.

In conclusion, HAL library could be a better choice if one does not need a lot of samples and tutorials.

### Program structure

The

Rtos or 前后台

## Algorithm evaluation

This section is to discuss the

### Kinematics algorithm

The kinematics algorithms used in this project are cite from current researches which have been approved many times. However, there is a precondition of using the inverse kinematics model, which is all the mechanical chains should have the same length, and the shape of the end effector is the same as the base. Besides, there is no limit on using trilateration, which means

Further work,

是新的方向

### Motor driver algorithm

The principle of control a stepper motor is to adjust the frequency of the impulse signal, in other words is to update the time delay between pull-up and pull-down operation. It is a complex problem of selecting the proper delay method on microcontroller. The method that is widely explained on tutorials is to use the delay function, however the delay function is realized with system tick timer (Systick) which can only be used in the main loop. The method used in Marlin firmware is to set a timer interrupt, the function of the interrupt routine is to do the IO operation and determine the delay before the next interrupt occur, but such method have to be used with Leibramp algorithm (Eiderman, 1994).

The method used in this project is to generate PWM with a general-purpose timer, and use another one as the monitor. The advantage is that

The disadvantage is that a feedback circuit have to be built and another IO is needed to capture the PWM, besides each stepper motor needs an individual timer so

In conclusion, the best solution for commercial purpose the

### Trajectory planning algorithm

As is explained in section 3.3.8, the trajectory is stored in a triple-pointers array. Such method benefits to the debugging process, because

On the other hand, the triple-pointers array increases the space complexity, because the blocks will no be generated until all the points on the path are found. Moreover, traversal is used on every trajectory operation which increases time complexity.

The data structure does affect the algorithm used on planning arc trajectory, for example one of the ideas from Bresenham’s circle algorithm is to divide the circle in 8-parts each octant of 45 degree

### Bresenham’s algorithm

The application of Bresenham’s algorithm in this project is quite straight forward which is to determine the remainder after division. However, there is a precondition of using this algorithm which is the gridding resolution must be 1. As is mentioned in the research part, the resolution of most of the home 3D printers is 0.1mm, so that this algorithm can be applied by multiplying 10 to the dividend, but it will fail when the resolution is improved to 0.05mm.

# Conclusion

This project is aimed to

The achievements include

[xc,yc,zc]=cylinder(2.5);

hold on;

[x,y,z]=cylinder(2.5);

z(2,:)=200;

surf(x-130,y,z,'FaceColor',[0,0,1]);

hold on;

surf(x+130,y,z,'FaceColor',[0,0,1]);

hold on;

surf(x,y+130,z,'FaceColor',[0,0,1]);

hold on;

syms x y z

xxx = zeros(1,10000000);

yyy = zeros(1,10000000);

zzz = zeros(1,10000000);

i = 0;

tic

syms x y z;

xxx = zeros(1,10000000);

yyy = zeros(1,10000000);

zzz = zeros(1,10000000);

n = 1;

z1 = 250;

while z1 <400

z2 = 250;

while z2<400

z3 = 250;

while z3<400

eq1 = (x-130).^2+(y).^2+(z-z1).^2-264^2;

eq2 = (x+130).^2+(y).^2+(z-z2).^2-264^2;

eq3 = (x).^2+(y+130).^2+(z-z3).^2-264^2;

s = solve(eq1,eq2,eq3,x,y,z);

aa = double(s.x);

bb = double(s.y);

cc = double(s.z);

aaa = aa(1,1);

bbb = bb(1,1);

ccc = cc(1,1);

if -90<aaa && aaa<90 && -90<bbb && bbb<90 && 0<ccc

xxx(n) = aaa;

yyy(n) = bbb;

zzz(n) = ccc;

end

n = n + 1;

z3 = z3 + 5;

end

z2 = z2 + 5;

end

z1 = z1 + 5;

end

scatter3(xxx,yyy,zzz,2,'blue','filled');

xlabel('x');

ylabel('y');

zlabel('z');

xlim([-180 180]);

ylim([-180 180]);

zlim([0 600]);

toc

z1 = 261;

z2 = 146;

z3 = 211;

while i <= 115

if mod(i\*5/8,1)>0.5

z1 = z1 - 1;

z3 = z3 - 1;

end

z1 = z1 - 1;

z2 = z2 + 1;

eq1 = (x-130).^2+(y).^2+(z-z1).^2-264^2;

eq2 = (x+130).^2+(y).^2+(z-z2).^2-264^2;

eq3 = (x).^2+(y+130).^2+(z-z3).^2-264^2;

s = solve(eq1,eq2,eq3,x,y,z);

aa = double(s.x);

bb = double(s.y);

cc = double(s.z);

aaa = aa(1,1);

bbb = bb(1,1);

ccc = cc(1,1);

xxx(i+1) = aaa;

yyy(i+1) = bbb;

zzz(i+1) = ccc;

i = i + 1;

end