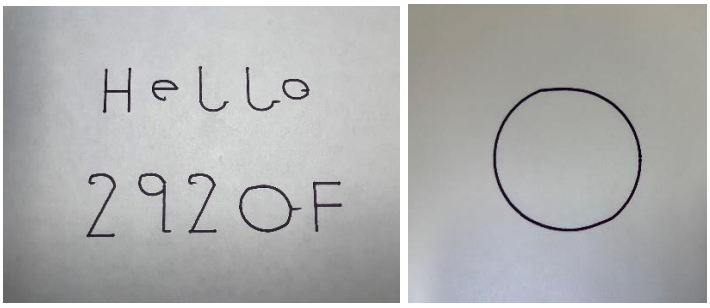
Project 3: 3D Hardware Digital Plotter

Name: Xiaocong Liu

1. The drawings



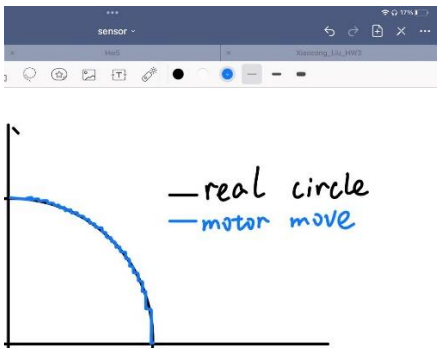
The drawings are the words “hello 2920F” and a circle with radius 3.0 cm. I attached the drawing videos in the folder named drawing record.

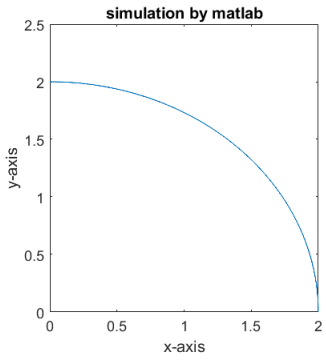
2. The most part of the difficult part in the project

The most difficult part I met was drawing a circle. My initial idea was to store the velocities of the x-axis ***Vsinθ*** and y-axis ***Vcosθ*** in two arrays respectively and use an index to track the current position. Then I would assign the value of TIM->ARR to the corresponding value in the array at that index. However, when I tried this approach, I found that the movements of these two axes weren't synchronized. So, my alternative solution was to compare the square of the distance from the current position to the center with the square of the radius to draw the circle, which I'll describe in detail in the next section.

2. The cleverest code segment of my system

The cleverest code for my system is how to draw a circle. The core of this method is to determine whether the current position is inside or outside the circle, thereby deciding whether to move one step along the x-axis or the y-axis. Taking the example of drawing a 1/4 circle in the first quadrant, if the distance from the current position to the center is less than the radius, $(X_i - X_0)^2 + (Y_i - Y_0)^2 < r^2$, it means it's inside the circle, and we need to move upward along the y-axis. Otherwise, it's outside the circle, $(X_i - X_0)^2 + (Y_i - Y_0)^2 \geq r^2$. We need to move one step to the left along the x-axis. Because each step of the motor is small enough that all x-axis and y-axis movements can form a circle. The process of drawing circles in the remaining 3 quadrants is the same as in the first quadrant, except that the x or y axes move in different directions. There is a simple graph to show the drawing process in the first quadrant.





The calculating process can also be optimized, assuming the center of the circle is (0,0) and the 1/4 circle is in the first quadrant. The distance formula for position i is

$$D_i = {X_i}^2 + {Y_i}^2 - r^2$$

if $D_i \geq 0$, move the point along x-axis one step and the distance formula is

$$D_{i+1} = (X_i - onemove_{stepmotor})^2 + {Y_i}^2 - r^2 = {X_i}^2 + {Y_i}^2 - r^2 - 2X_i + {onemove_{stepmotor}}^2$$

$$D_{i+1} = D_i - 2X_i + {onemove_{stepmotor}}^2$$

if $D_i < 0$, move the point along y-axis one step and the distance formula is summarized by using the same method.

$$D_{i+1} = D_i + 2Y_i + {onemove_{stepmotor}}^2$$

After having these two equations, the steps to move x-axis or y-axis becomes checking the value of D and updating x or y.

$$if\ D \geq 0 \qquad D_{i+1} = D_i - 2X_i + {onemove_{stepmotor}}^2$$

$$X_{i+1} = X_i - onemove_{stepmotor}$$

$$if\ D < 0 \qquad D_{i+1} = D_i - 2X_i + {onemove_{stepmotor}}^2$$

$$Y_{i+1} = Y_i + onemove_{stepmotor}$$

The optimized approach could also be applied in the remaining three quadrants. Therefore, the computation complexity for calculating the distance from the current position to the center has been simplified, and it requires only one variable to record the value of D, thus saving microcontroller memory resources.

In summary, this method is easy to implement and does not require significant computational or storage resources. Additionally, it achieves a reasonably high precision in drawing the circle.