DEVELOPMENT OF ARDUINO CODE FOR SCARA ROBOTIC ARM

Alekh Kumar Sinha[1]

[1]*Tata consultancy Services*

Abstract

This paper deals with the development of arduino code for SCARA (Selective compliance assembly robotic arm). It accepts G-Code and converts Cartesian coordinates to polar coordinates and uses those to move arms to that coordinate. It basically modifies CNC firmware (working on Cartesian coordinates) available in github to suit for polar coordinates.

----------***----------

1. INTRODUCTION

I work in Tata Consultancy Services. There they have excellent initial learning program (ILP) which they give to fresher. There they encourage trainee to make innovative project. My project was development of SCARA arm.

2. CNC FIRMWARE

We downloaded CNC firmware from GITHUB website and used Arduino to load those codes. Main problem with this code was that, it was made to take Cartesian Coordinates as input, but our requirement was polar coordinates. A separate function was added to convert Cartesian Coordinates to polar coordinates and also a new function was added to calculate steps. It calculated angles that SCARA required to move for achieving those coordinates, by using inverse kinematics. Depending on the stepper motor configuration, those angles were converted to steps, which were given as input to the stepper motor. Stepper motor has to move by those steps to achieve those coordinates.

Other changes are in extracting coordinates from G-codes command. We converted whole command to string, then used inbuilt functions to extract numbers from those commands. **Bresenham's line** algorithm was used to move two stepper motor together.

The next task was to take input from the notepad. For that we used python shell and downloaded MYSERIAL module and followed instruction given on http://www.meccanismocomplesso.org/en/arduino-tutorial-serial-data-actuator/

3. ABOUT THE HARDWARE

Arduino was used to integrate logic with the hardware. Arduino is a micro controller with various output and input pins. We used POLOLU 1.4 ramps board to move stepper motor together. Through Arduino interface one can load logic into the controller and define output pins. Arduino programming can be basically divided into three blocks.

- Defining of pins and initialization of variable
- Setup block loop block

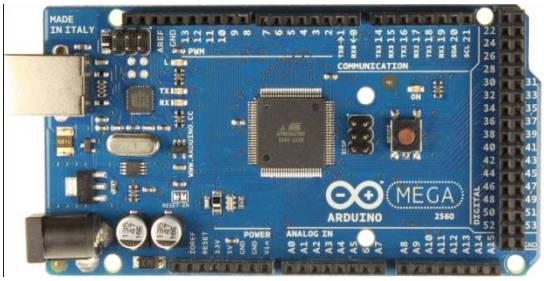


Fig 1: Arduino Mega

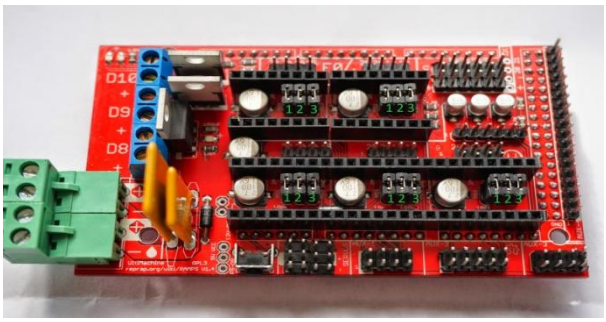
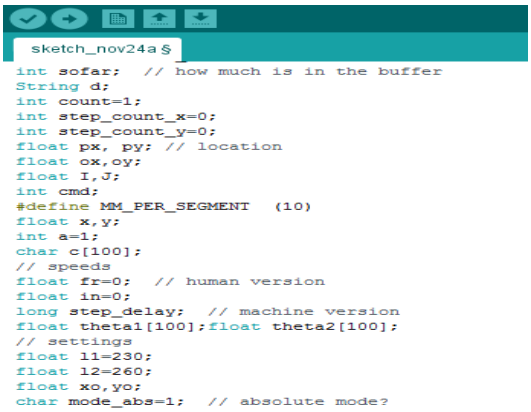


Fig 2: Pololu Ramp 1.4

3.1 Defining Pins and Initialization of Variables

In this block pins are defined. Syntax is #define variable_namepin_number. All the pins are defined here. Initialization of variable is also done here. Arduino supports dynamic initialization but global variables definition needs to be done here only. Syntax for this is data_typevariable_name value.


sketch_nov24a §

```
int sofar;  // how much is in the buffer
String d;
int count=1;
int step_count_x=0;
int step_count_y=0;
float px, py; // location
float ox,oy;
float I,J;
int cmd;
#define MM_PER_SEGMENT  (10)
float x,y;
int a=1;
char c[100];
// speeds
float fr=0;  // human version
float in=0;
long step_delay;  // machine version
float theta1[100];float theta2[100];
// settings
float l1=230;
float l2=260;
float xo,yo;
char mode_abs=1;  // absolute mode?
```

Fig 3: Variable declaration

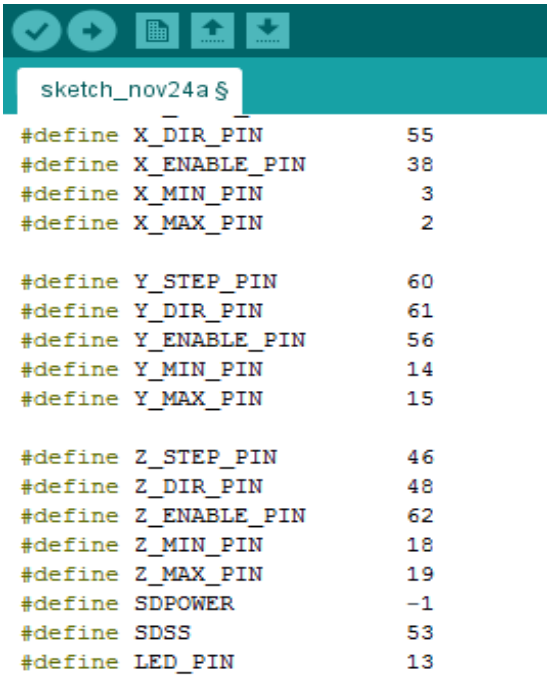


Fig 4: Defining of pins

3.2 Setup Block

This block deals with the output and input pin definitions. In this program position() function is called. In position function initial coordinates of the end collector is taken as input. This data is used to determine the initial angular position of both the link. The initial angular position of link1 and link2 is stored in the array theta1[0] and theta2[0]. Further angular positions are also stored in this array and the value is compared with initial position to calculate the angle by which link needs to rotate

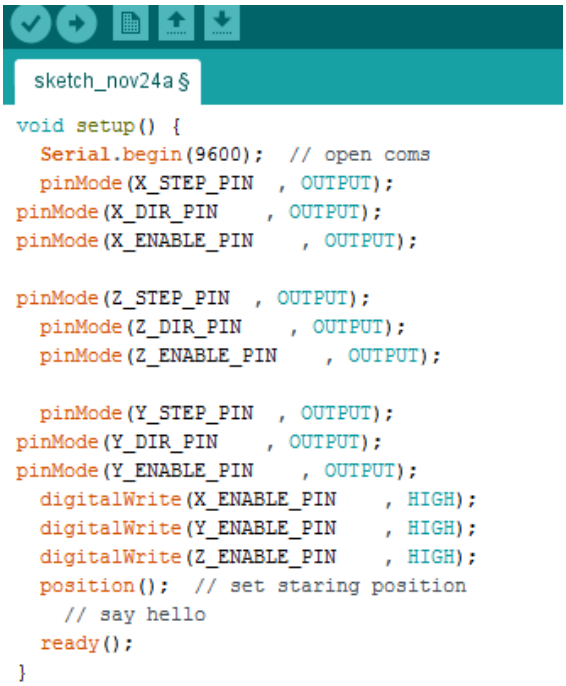


Fig 5: Setup function

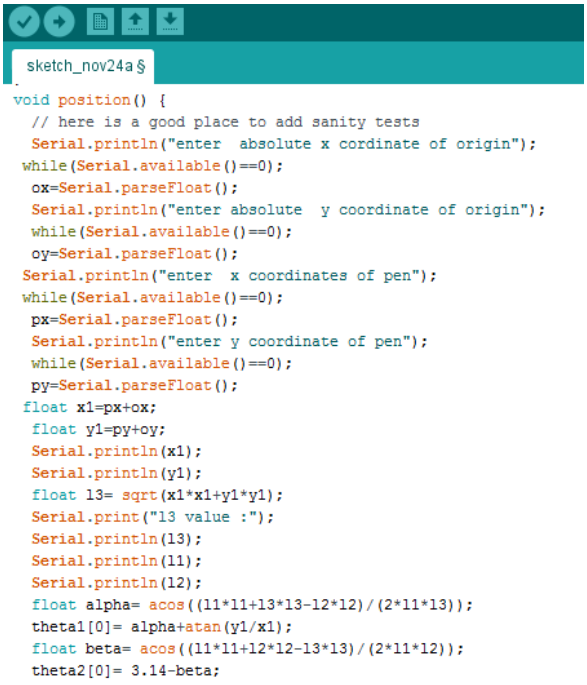


Fig 6: Position function

4. CONTROL LOGIC

Control logic can be divided into following steps:-

- Extraction of coordinates from the CAD model
- Take G-codes as input
- Extract coordinates from the G-Code
- Convert Cartesian coordinates to polar coordinates
- Calculation of angles by which links need to rotate
- Convert those angles to steps
- Move stepper motor by those steps

4.1 Extraction of Coordinates from the G-Code

For extraction of coordinates from G-Code, whole G-Code command was converted into string and coordinates were extracted from the command by using library functions. Number which is preceded with G in the G code determines the type of command.

Fig 7: Part of code showing extraction of coordinates

As you can see in the figure 6 picture, the first mark highlights command to convert whole G-Code statements to string. String.read() function is used for this purpose. This is a library function which converts everything which is read as input to string. 'd' is the variable of data type string and it stores the G-Code command as string.

The second oval mark extracts number, which is after G in the G-code command. This number is extracted by using toInt() command. The syntax for this command is string.toInt(). As number needs to be extracted from the part of the string which is after G, d.substring(i+1) is used along with toInt() function. (i+1) denotes the subscript of the character which is after G.

Similarly, number after X is extracted from the G codes. This is marked by the third mark. toFloat() function is used for this purpose. Last box denotes extraction of number which is after letter Y in the G-Code command. Other coordinates such as Z can be extracted with similar logic.
These x coordinates and y coordinate are with respect to the origin of the work piece but we want all the coordinate with respect to point of rotation of the first link so, origin coordinate with respect to point of rotation of the first link is asked as input and is added to all the coordinates extracted from the G-code, which converts all coordinates with respect to point of rotation of the first link.

4.2 Angle Calculation

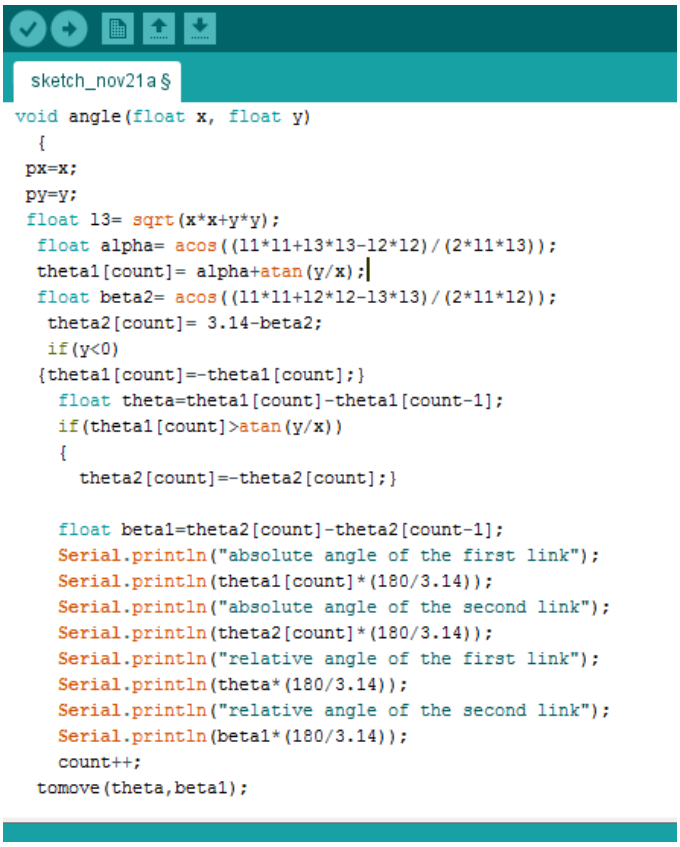


Fig 8: Part of Code showing angle calculation

A user-defined function angle was created which accepts coordinates and calculates angles by which both the link needs to rotate. This function was called from the loop block.

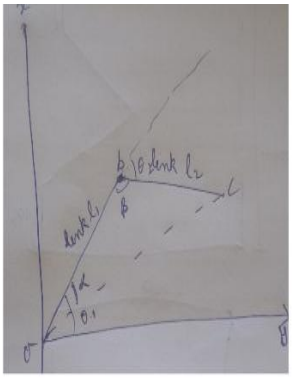
If G01 is used with G90 then whatever coordinates extracted is sent by adding origin which converts those to absolute coordinates, but if G91 is used then absolute coordinates of the previous position needs to be added to the extracted coordinates to generate absolute coordinates of the new position. Here px and py stores absolute coordinate x and y coordinates of the previous position.

For angle calculation inverse kinematics is used

4.2.1 Inverse Kinematics

Inverse kinematics is used to get polar coordinates from the Cartesian coordinatesAs you can see in the figure 8 if we join o with c then it will form a triangle. We know the length all the sides of the triangle, every angle can be calculated using Cosine law.

Let L3 be an imaginary link connecting ac.
With that link l1 , link l2 and link l3 forms a triangle.
Length of $L_3=\sqrt{x^2+y^2}$
Now on application of cosine rule we can say
$\alpha=cos^{-1}\frac{l_1^2+l_3^2-l_2^2}{2*l_1l_3}$
By referring figure $\emptyset_1=\alpha+tan^{-1}\frac{y}{x}$
Similarly $\beta=cos^{-1}\frac{l_1^2+l_2^2-l_3^2}{2*l_1l_2}$
$\emptyset_2=180-\beta.$



Input To the 1st stepper motor- /(360/number of step in one revolution)
Input To the 2nd stepper motor- /(360/number of step in one revolution)

Fig 9: Inverse Kinematic Calculation

4.3 G02 and G03 Command

G03 and G04 is used for tracing arc. This has five input preceded by letters G,X,Y,I and Jrespectively. Number after X and Y are x and y coordinate of the end point of arc and I and J are vertical and horizontal increment respectively.

sketch_nov21a §

```
void arc(float cx,float cy,float x,float y,float dir) {
  float dx = px - cx;
  float dy = py - cy;
  float radius=sqrt(dx*dx+dy*dy);
  float angle1=atan3(dy,dx);
  float angle2=atan3(y-cy,x-cx);
  float theta=angle2-angle1;
  if(dir>0 && theta<0) angle2+=2*PI;
  else if(dir<0 && theta>0) angle1+=2*PI;
  theta=angle2-angle1;
  float len = abs(theta) * radius;

  int i, segments = ceil( len * MM_PER_SEGMENT );

  float nx, ny, angle3, scale;

  for(i=0;i<segments;++i) {
    // interpolate around the arc
    scale = ((float)i)/((float)segments);

    angle3 = ( theta * scale ) + angle1;
    nx = cx + cos(angle3) * radius;
    ny = cy + sin(angle3) * radius;
    // send it to the planner
    angle(nx,ny);
  }
```

Fig 10: Part of coding showing interpolation function

Arc is divided into segments and is approximated to be formed of small lines. These inputs are used to interpolate arc to find intermediate points and the coordinates of those points are fed in the previously explained function angle. As shown in the above figure, arc tangent is found which gives the arc angle and that angle is divided into segments to find intermediate points. The coordinates of intermediate points is send as input it to the angle function and same as in the case of line, process is repeated.

4.4 To Move Stepper Motor Together

sketch_nov24a §

```
void tomove(float theta,float beta1) {
long dx=theta/(3.14/9600);
long dy=beta1/(3.14/9600);
int dirx=dx>0?1:-1;
int diry=dy>0?-1:1;  // because the motors are mounted in opposite directions
dx=abs(dx);
dy=abs(dy);

long i;
long over=0;

if(dx>dy) {
  over=dx/2;
  for(i=0;i<dx;++i) {
    m1step(dirx);
    over+=dy;
    if(over>=dx) {
      over-=dx;
      m2step(diry);
    }
    pause(step_delay);
  }
} else {
  over=dy/2;
  for(i=0;i<dy;++i) {
    m2step(diry);
    over+=dx;
```

Fig 11: Bresenham'salgorithm

In order to move stepper motor, those angles were converted to steps. We have used NEMA 17 which covers 200steps in one complete rotation. We have used POLUA RAMPS 1.4 BOARD. It has pins for connecting three motors and can make them work simultaneously. Jumper along with the stepper motor driver is used which enables micro- stepping .This micro stepping divides each step into 32 sub step so, in 360 deg of rotation it has 6400(32*200) micro steps. With this we can conclude that 1 degree can be covered by 3200/180 sub steps. We have used a belt transmission with diameter of the pulley in the ratio 1:3. This means that with 3deg of rotation of stepper motor, link will rotate by 1 deg. This can be further interpreted as for rotating link by one degree stepper motor needs to move by 9600/180 (3*3200/180) sub steps.

By using the above stated concept steps are calculated from angles. First few statement of the angle function (as shown in the figure) illustrate this concept.

The above calculated steps are fed in Bresenham's algorithm to move both the stepper motors together. Bresenham's algorithm ensures that both the motor complete their steps together. For example if a stepper motor needs to move by 50 steps and other by 100 steps ,their speed will be so adjusted that both the motor will complete their steps together.

4.5 Debugging and Validation

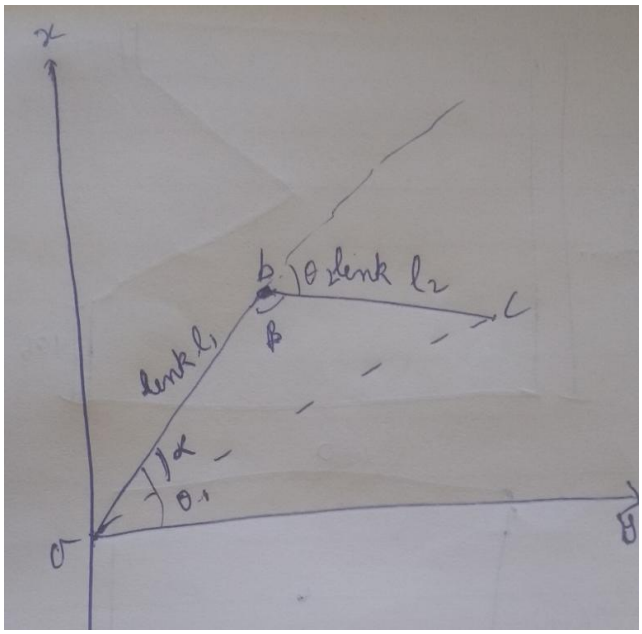


Fig 12: Kinematic Representation of SCARA arm

As you can see in the above figure, SCARA model can be simplified and can be represented by a kinematic line diagram. This line diagram has one re-volute pair at o and other at b. The end c represents the position of the end collector. Now we can draw this line diagram in a designing software and utilize this concept to find angles by which links need to rotate.

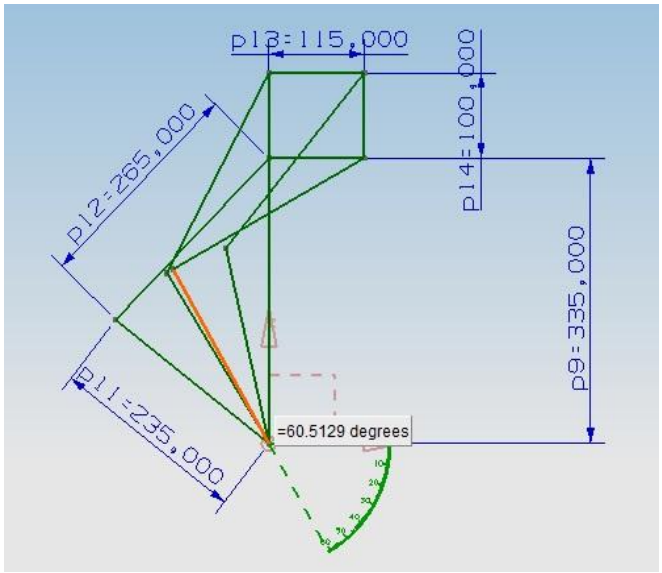


Fig 13: line diagram drawn in designing software to find out angles that links require to move

We need to decide at what coordinate we want the end collector to be. With this decision, point b is fixed and now a line is drawn whose length is equal to the length of link 1 and other whose length is equal to the length of the link 2 (as shown in the figure). With the help of this diagram we can calculate the angle between link1 and x axis and also angle between link 2 and link1 This angle will change when different points are selected for b. The difference between those angles will give angle by which links needs to rotate to achieve those coordinate.

For example at point angle between link 1 and x axis is 141 deg and in point B it changes to 118 deg so link 1 needs to move by 22 deg for moving end collector from position A to position B.

This can be used for debugging and also for validation of the codes. As shown in the figure below ,the value which is sent to 'tomove' function is printed. This value represents the angle by which stepper motor will move. A step counter is also included in step function to calculate how many steps stepper motor has moved. If these things tally with the design software generated data then program is correct or else its require debugging.

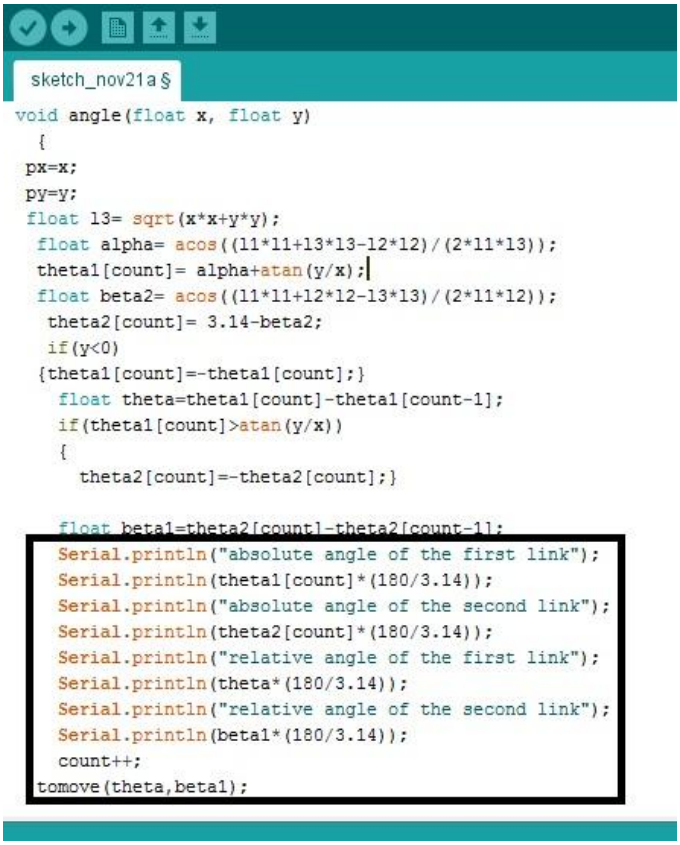


Fig 14: Angle output to be validated by line diagram

Table 1: Data extracted from CAD model

Sl no	Coordinates	Absolute angles moved by link 1	Absolute angles moved by link 2	Change in in angle for link 1	Change in angle for link 2	Direction
1	(0,335)	141.8864	96.09			
2	(115,335)	119.48	90	22.40	6.09	Anticlockwise,clockwise
3	(115,435)	102.76	51.33	16.72	38.67	Anticlockwise,clockwise
4	(0,435)	121.55	59.2	18.79	7.87	Anticlockwise,clockwise

Table 2: Data extracted from software

Sl no	coordinates	Angle required to move	Angle it actually moved
1	(0,335)		
2	(115,335)	22.40 6.09	22.51 6.33
3	(115,435)	16.72 38.67	18.15 40.89
4	(0,435)	18.79 7.87	19.2 8.175

5. CONCLUSION

Scara is very useful as its very compact. This paper will help in developing control logic for SCARA model. This model has varied application in case of pick and place robot, or automated spot welding etc.

REFERENCES

[1] Codes from GITHUB
[2] https://www.arduino.cc/en/Tutorial/HomePage
[3] https://en.wikipedia.org/wiki/SCARA
[4] https://github.com/grbl/grbl