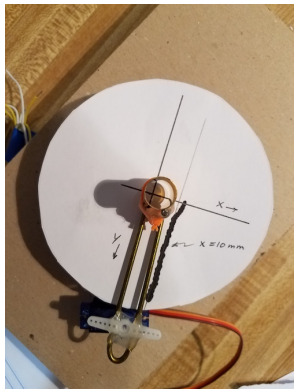
In [126]:
```
#send a string of characters directly
import serial
ser = serial.Serial('/dev/tty.usbmodem143141')

ser.write(b'255,10,254,0')
ser.close()
```

Code above sends a sting to this machine. The machine is documented in https://roberthart56.github.io/SCFAB/SC_lab/PS70_machine/index_rev1.html (https://roberthart56.github.io/SCFAB/SC_lab/PS70_machine/index_rev1.html)



In [ ]:
```
#write string from array of numbers, one number at a time.

import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')

array_1 = np.array([255,10])

for index in [0,1]:
    sring=str(array_1[index])
    ser.write(sring.encode())
    ser.write(b',')
    print(sring)
ser.close()
print(array_1)
```

In [ ]:
```
'''
Expand to four elements.
Put string together from stringifying four array elements, with commas.
This works.

'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')

array_1 = np.array([255,10, 254,30])
string_to_send = ""
for index in range(4):
    string_to_send += str(array_1[index])
    if (index<3):
        string_to_send += ","

ser.write(string_to_send.encode())
ser.close()

print('string_to_send',string_to_send)

print(array_1)
```

In [9]:
```
'''
Now put string together from arrays of steps and angles.
In the form: 255, steps, 254, angle

'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
step_array = np.array([5,5,-5,-5])
angle_array = np.array([0,5,10,15])

for ind_1 in range(4):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(1)
    print('string_to_send',string_to_send)

ser.close()
```

```
string_to_send 255, 5 ,254, 0
string_to_send 255, 5 ,254, 5
string_to_send 255, -5 ,254, 10
string_to_send 255, -5 ,254, 15
```

In [35]:
```
#This is working out the details of calculating stepp array from theta array.
#includes rounding and turning into integer array.

import numpy as np

theta_array = np.array([0,15,30,45])
step_array = np.zeros(theta_array.size)    #initialize array for steps.


step_array[0] = 0
for i in range(1,theta_array.size):
    step_array[i] = (theta_array[i] - theta_array[i-1])*200/360

step_array = (np.round(step_array))
step_array = step_array.astype(int)

print(step_array)
```

```
[0 8 8 8]
```

In [41]:
```
#work out converting r to servo angle.
import numpy as np
R = 10
r_array = np.array([2,2.5,3.0,4.0])     # r is radius to be converted to servo angle.
angle_array = np.arcsin(r_array/R)*(360/2/np.pi)
angle_array = np.round(angle_array)
angle_array = angle_array.astype(int)  #here's the step array to send to machine.

print(angle_array)
```

```
[12 14 17 24]
```

In [44]:
```
'''
Now derive the steps and angles from arrays of:
theta:  to be translated into steps
r:  to be turned into a servo angle. (this would be exact
if R of servo arm were infinite.)

'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
R = 5.8    #mm radius of servo arm.
theta_array = np.array([0,15,30,45])    #theta corresponds to angle between point and y-axis.
r_array = np.array([2,2.5,3.0,4.0])     # r is radius to be converted to servo angle.

step_array = np.zeros(theta_array.size)    #initialize array for steps.
angle_array = np.zeros(theta_array.size)    #initialize array for servo angles.

#Now work on step array
step_array[0] = 0
for i in range(1,theta_array.size):
    step_array[i] = (theta_array[i] - theta_array[i-1])*200/360

step_array = (np.round(step_array))
step_array = step_array.astype(int)  #here's the step array to send to machine.

#Now work on angle array

angle_array = np.arcsin(r_array/R)*(360/2/np.pi)
angle_array = np.round(angle_array)
angle_array = angle_array.astype(int)  #here's the step array to send to machine.

for ind_1 in range(4):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(1)
    print('string_to_send',string_to_send)

ser.close()
```

```
string_to_send 255, 0 ,254, 20
string_to_send 255, 8 ,254, 26
string_to_send 255, 8 ,254, 31
string_to_send 255, 8 ,254, 44
```

In [82]:

```
#work out array arithmnetic for converting x,y to r, theta, x,y >0

import numpy as np

R = 58    #mm radius of servo arm.


x_array = np.array([10,10,10,10])
y_array = np.array([5,15,25,35])
array_size = x_array.size

theta_array = np.arctan(y_array/x_array)    #theta corresponds to angle between point and x-axis in radians.
gamma_array = 90 - theta_array*360/2/np.pi    #gamma is the angle through which the stepper steps to get to the point.

r_array = np.sqrt((x_array)**2 + y_array**2)     # r is radius to be converted to servo angle.

print(theta_array)
print(gamma_array)

print (r_array)
print(array_size)
```

```
[ 0.46364761  0.98279372  1.19028995  1.29249667]
[ 63.43494882  33.69006753  21.80140949  15.9453959 ]
[ 11.18033989  18.02775638  26.92582404  36.40054945]
4
```

In [66]:
```
'''
Now start with x and y arrays.  Assume R is infinite, so that calculation
is same as for regular polar coordinates.


'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
R = 58    #mm radius of servo arm.

x_array = np.array([10,10,10,10])
y_array = np.array([5,15,25,35])
array_size = x_array.size

theta_array = np.arctan(y_array/x_array)    #theta corresponds to angle between point and x-axis in radians.
gamma_array = 90 - theta_array*360/2/np.pi    #gamma is the angle through which the stepper steps to get to the point.

r_array = np.sqrt((x_array)**2 + y_array**2)     # r is radius to be converted to servo angle.


step_array = np.zeros(array_size)    #initialize array for steps.
angle_array = np.zeros(array_size)    #initialize array for servo angles.

#Now work on step array
step_array[0] = 0
for i in range(1,array_size):
    step_array[i] = (gamma_array[i] - gamma_array[i-1])*200/360

step_array = (np.round(step_array))
step_array = step_array.astype(int)  #here's the step array to send to machine.

#Now work on angle array

angle_array = np.arcsin(r_array/R)*(360/2/np.pi)
angle_array = np.round(angle_array)
angle_array = angle_array.astype(int)  #here's the step array to send to machine.

for ind_1 in range(4):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(1)
    print('string_to_send',string_to_send)

ser.close()
```

```
string_to_send 255, 0 ,254, 11
string_to_send 255, -17 ,254, 18
string_to_send 255, -7 ,254, 28
string_to_send 255, -3 ,254, 39
```

Markdown cell.

In [88]:
```
'''
Now start with x and y arrays.  Assume R is infinite, so that calculation
is same as for regular polar coordinates.


'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
R = 58    #mm radius of servo arm.

x_array = 10*np.ones(40)
y_array = y= np.arange(0,40)
array_size = x_array.size

theta_array = np.arctan(y_array/x_array)    #theta corresponds to angle between point and x-axis in radians.
gamma_array = 90 - theta_array*360/2/np.pi    #gamma is the angle through which the stepper steps to get to the point.

r_array = np.sqrt((x_array)**2 + y_array**2)     # r is radius to be converted to servo angle.


step_array = np.zeros(array_size)    #initialize array for steps.
angle_array = np.zeros(array_size)    #initialize array for servo angles.

#Now work on step array
step_array[0] = 0
for i in range(1,array_size):
    step_array[i] = (gamma_array[i] - gamma_array[i-1])*200/360

step_array = (np.round(step_array))
step_array = step_array.astype(int)  #here's the step array to send to machine.

#Now work on angle array

angle_array = np.arcsin(r_array/R)*(360/2/np.pi)
angle_array = np.round(angle_array)
angle_array = angle_array.astype(int)  #here's the step array to send to machine.

for ind_1 in range(array_size):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(2)
    print('string_to_send, number', ind_1,":",string_to_send)

ser.close()
```

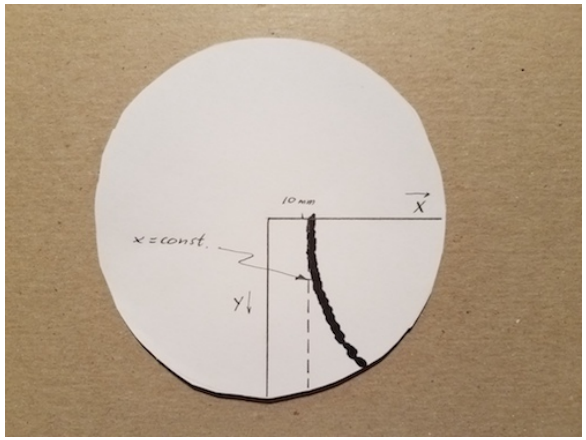
```
string_to_send, number 0 : 255, 0 ,254, 10
string_to_send, number 1 : 255, -3 ,254, 10
string_to_send, number 2 : 255, -3 ,254, 10
string_to_send, number 3 : 255, -3 ,254, 10
string_to_send, number 4 : 255, -3 ,254, 11
string_to_send, number 5 : 255, -3 ,254, 11
string_to_send, number 6 : 255, -2 ,254, 12
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-88-bf4b367dcdaa> in <module>()
     45     string_to_send += str(angle_array[ind_1])
     46     ser.write(string_to_send.encode())
---> 47     time.sleep(2)
     48     print('string_to_send, number', ind_1,":",string_to_send)
     49

KeyboardInterrupt:
```

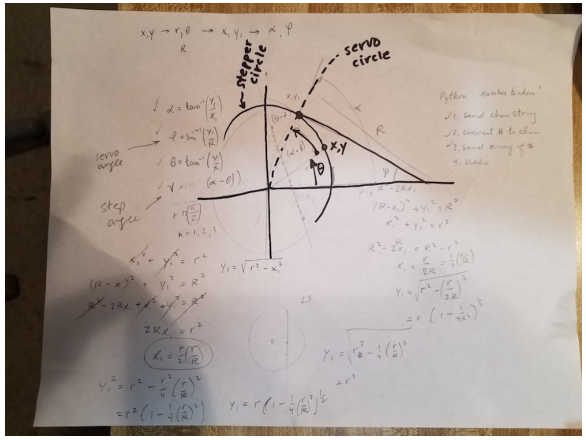
In [ ]:
```
'''
Now start with x and y arrays.  Assume R is infinite, so that calculation
is same as for regular polar coordinates.

Adjust method of rounding step size. First round, then take deltas, so that steps don't go to zero.

See output for this program in fig01.  x=10mm line is approximated by a curve due to finite radius of
servomotor arm.

'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
R = 58    #mm radius of servo arm.

x_array = 10*np.ones(40)
y_array = np.arange(0,40)
array_size = x_array.size

theta_array = np.arctan(y_array/x_array)    #theta corresponds to angle between point and x-axis in radians.
gamma_array = 90 - theta_array*360/2/np.pi    #gamma is the angle through which the stepper steps to get to the point.

r_array = np.sqrt((x_array)**2 + y_array**2)     # r is radius to be converted to servo angle.


step_array = np.zeros(array_size)    #initialize array for steps.
angle_array = np.zeros(array_size)    #initialize array for servo angles.

#Now work on step array
gamma_array = gamma_array - gamma_array[0]    #start at first angle
total_step_array = gamma_array*200/360    #convert to total steps.
total_step_array = np.round(total_step_array)  #round to integers.

step_array[0] = 0
for i in range(1,array_size):
    step_array[i] = total_step_array[i] - total_step_array[i-1]   #takes delta between total step values.


step_array = step_array.astype(int)  #here's the step array to send to machine.

#Now work on angle array

angle_array = np.arcsin(r_array/R)*(360/2/np.pi)
angle_array = np.round(angle_array)
angle_array = angle_array.astype(int)  #here's the step array to send to machine.

for ind_1 in range(array_size):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(2)
    print('string_to_send, number', ind_1,":",string_to_send)

ser.close()
# print(x_array)
```

```
# print(y_array)
# print(theta_array)
# print(gamma_array)
# print(step_array)
```

The code above calculates a curve for the line x=10mm from y=0 to y=40 mm. Since the servo arm is not infinite, the servo angle does not translate exactly to radial distance from center. Rather than plotting a straight line for x=const, the line curves away from the y-axis, as expected.



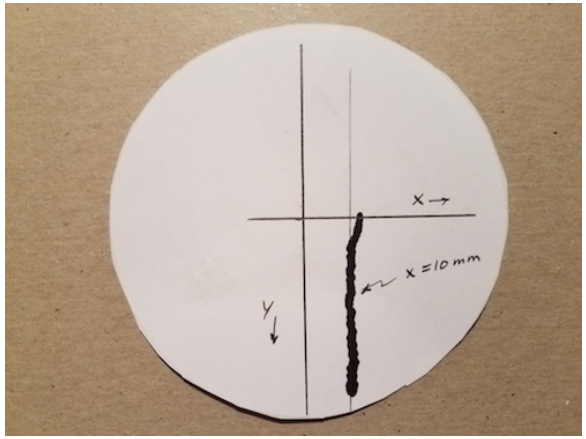
Here is a sketch of the setup, showing the geometry used to calculate the proper servo angle and step number for a given x,y point. The code below incoporates these calculations.



In [ ]:
```
'''
Startingg from program above, add calculations for finite arm.  See sketch of page above with algebra!  This is a mess, and
will probably contain errors! Actually not as many as expected.  With some minor edits, it works.  See below.
'''
import serial
import time
import numpy as np

ser = serial.Serial('/dev/tty.usbmodem143141')
R = 58    #mm radius of servo arm.

x_array = 10*np.ones(40)
y_array = np.arange(0,40)
array_size = x_array.size
step_array = np.zeros(array_size)    #initialize array for steps.
angle_array = np.zeros(array_size)    #initialize array for servo angles.


theta_array = np.arctan(y_array/x_array)    #theta corresponds to angle between point and x-axis in radians.
r_array = np.sqrt((x_array)**2 + y_array**2)     # r is radius to be converted to servo angle.

x_1_array = r_array**2/2/R   #x1 is the x-coord of the desired intersection point of the two circles.
y_1_array = np.sqrt(r_array**2-x_1_array**2)    #y1 is the x-coord of the desired intersection point.

alpha_array = np.arctan(y_1_array/x_1_array)    #alpha corresponds to angle between intersection point and x-axis in radians.


gamma_array = (alpha_array - theta_array)*360/2/np.pi    #gamma is the angle through which the stepper steps to get to the point.

#Now work on step array
gamma_array = gamma_array - gamma_array[0]    #start at first angle
total_step_array = gamma_array*200/360    #convert to total steps.
total_step_array = np.round(total_step_array)  #round to integers.

step_array[0] = 0
for i in range(1,array_size):
    step_array[i] = total_step_array[i] - total_step_array[i-1]   #takes delta between total step values.


step_array = step_array.astype(int)  #here's the step array to send to machine.

#Now work on angle array, for servomotor arm.

angle_array = np.arcsin(y_1_array/R)*(360/2/np.pi)    # servo angle absed on calculated intersection of two circles.
angle_array = np.round(angle_array)

angle_array = angle_array.astype(int)  #here's the step array to send to machine.

for ind_1 in range(array_size):
    string_to_send = "255, "
    string_to_send += str(step_array[ind_1])
    string_to_send += " ,254, "
    string_to_send += str(angle_array[ind_1])
    ser.write(string_to_send.encode())
    time.sleep(2)
    print('string_to_send, number', ind_1,":",string_to_send)

ser.close()
```

```
#print(x_array)
# print(y_array)
# print(r_array)
# print(theta_array)
# print(x_1_array)
# pri
# nt(y_1_array)
# print(gamma_array)
# print(step_array)
```

Here is the line plotted by the code above. Some wiggles for small y-values, but mostly corrected.



The plotting takes two seconds per point. I have put generous pauses almost everywhere in the code for the machine. The next steps could include working out the timing to speed things up. Or not.

In [ ]: