Pseudocode:

```
1. Identify the graph line
    gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    kernel = np.ones((2,2),np.uint8)
    opening = cv2.morphologyEx(thresh)

    _, contours, hierarchy =
cv2.findContours(opening,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)

   return contours

2. Get the coordinates of points on the graph line

cnt = max(find_contours(img), key=cv2.contourArea)
cv2.drawContours(mask, [cnt], 0, 255, -1)

res = cv2.bitwise_and(img, img, mask=mask)

non_zero = cv2.findNonZero(res)

for i in range(0, len(non_zero))
    first_x = non_zero[i][0][0]
    first_y = non_zero[i][0][1]
    first = res[first_y, first_x]
    res[first_y, first_x] = 255

for contour in find_contours(res)
    cv2.drawContours(mask2, [contour], 0, 255, -1)
    res2 = cv2.bitwise_and(res,res,mask=mask2)
    i = i+1

indices = np.where(mask2 == [255])
coordinates = zip(indices[0], indices[1])

3. Use some sort of regression to find the equation of the graph from these points

  degree = 2
  coefs, res, _, _, _ = np.polyfit(x,y,degree, full = True)
  ffit = np.poly1d(coefs)
  print (ffit)
```

The client I identified is Pada Schafnner. Pada is a Junior at the Dwight school. I identified him as a client because he is particularly passionate about math. While working on a physics lab report, Pada noted that he wished there was a way he could find the equation of a graph easily. I set up an in-person interview with Prada by contacting him via Snapchat. I asked him what kind of information he would like from the graph. He said all he wanted to know was a basic equation that described the behavior of the graph. I asked him how he wanted the program to function. Pada proceeded to tell me it would be useful to be able to take a photo of a graph and know its equation from that. After our interview, I got to pseudo coding. I broke the problem up into a couple steps:

1. Identify the graph line
2. Get the coordinates of points on the graph line
3. Use some sort of regression to find the equation of the graph from these points

For step 1, I pseudocoded a series of image preprocessing functions. I dilated then eroded the image, then and ran a Otsu Threshold over image. I thought that this should have been good enough to find the graph line (however, as I learned during my implementation, I was wrong). For step 2, I simply looped through all the pixels on the image, storing the coordinates of only the white pixels (the graph line was converted to all white pixels by the Otsu Threshold in step 1. Finally, for step 3, I found that a polynomial curve fitting algorithm would be adequate. When I actually implemented my pseudocode, I ran into some issues however. As I learned, my preprocessing on the image was not enough to extract the graph line. Once I had denoised the image (by dilation and erosion), and then ran an Otsu Threshold on the image, the graph line was still not found. I had to repeat these steps again in order to get rid of the grid lines. Once I did this, the graph line was found. Implementing steps 2 and 3 were pretty straight forward. Once I completed the prototype, I had Pada test it out. He said that the program worked ok, but the equation found was not always accurate. Pada also said he wished the program could work on an iPhone so that he could run it in real time. For Padas first piece of feedback, I changed the degrees of the polynomial curve fitting equation. This made the equation found slightly more accurate. For Padas second piece of advice, I was unable to find a solution in time. Improving my solution is definitely something I will continue to do in the future, however. Over all I think my final solution turned out well. Pada said that it worked adequately.