Appendix A. Python Code

The full working repository can be found at <https://github.com/lofro/TIE_OCR>. The code consists of three main files “Main.py”, “Img\_to\_Word.py”, “serial\_rx\_tx.py”. The Main.py creates the user interface and connect the user’s interactions to other systems. Img\_to\_Word.py converts a collected frame into an output of a desired string. Serial\_rx\_tx.py creates a serial port in python, it also handles the communication on the back end between the python interface and the serial output.

Appendix A.1. Main.py

import cv2

import tkinter #python quick user interface

import tkinter.scrolledtext as tkscrolledtext

import PIL.Image, PIL.ImageTk

import time

import sys

import datetime

import threading

import queue

#My libs

import serial\_rx\_tx

import Img\_to\_Word

#Initialize serial out port (will be COM3 at 9600 but can be changed)

serialPort = serial\_rx\_tx.SerialPort()

#Initialize log file

logFile = None

#Main app class

class App:

    # Inital code steps up all the user interface

    def \_\_init\_\_(self, window, window\_title, video\_source=0):

        self.thread\_queue = queue.Queue() #queue for thread

        self.window = window # tkinter

        self.window.title(window\_title)

        self.video\_source = video\_source

        self.outtext="Start"

        # open video source (by default this will try to open the computer webcam)

        self.vid = MyVideoCapture(self.video\_source)

        # Create a canvas that can fit the above video source size

        self.canvas = tkinter.Canvas(window, width = self.vid.width, height = self.vid.height)

        self.canvas.grid(row=0,column=0, rowspan=10)

        # Comport

        self.label\_comport = tkinter.Label(window,width=8,height=2,text="COM Port:")

        self.label\_comport.grid(column=1, row=0)

        self.label\_comport.config(font="bold")

        #COM Port entry box

        self.comport\_edit = tkinter.Entry(window,width=8)

        self.comport\_edit.grid(column=2, row=0)

        self.comport\_edit.config(font="bold")

        self.comport\_edit.insert(tkinter.END,"COM3")

        # COM Port open/close button

        self.button\_openclose = tkinter.Button(window,text="Open COM Port",width=15,command=self.OpenCommand)

        self.button\_openclose.config(font="bold")

        self.button\_openclose.grid(column=3, row=0)

        #Clear Rx Data button

        self.button\_cleardata = tkinter.Button(window,text="Clear Rx Data",width=15,command=self.ClearDataCommand)

        self.button\_cleardata.config(font="bold")

        self.button\_cleardata.grid(column=4, row=0)

        #Baud Rate label

        self.label\_baud = tkinter.Label(window,width=8,height=2,text="Baud Rate:")

        self.label\_baud.grid(column=1, row=1)

        self.label\_baud.config(font="bold")

        #Baud Rate entry box

        self.baudrate\_edit = tkinter.Entry(window,width=8)

        self.baudrate\_edit.grid(column=2, row=1)

        self.baudrate\_edit.config(font="bold")

        self.baudrate\_edit.insert(tkinter.END,"9600")

        #Refresh label

        self.label\_refresh = tkinter.Label(window,width=15,height=2,text="Refresh Rate:")

        self.label\_refresh.grid(column=3, row=1)

        self.label\_refresh.config(font="bold")

        #Picture refresh button

        self.button\_senddata = tkinter.Button(window,text="Refresh Rate (s)",width=15,command=self.Refresh\_Rate)

        self.button\_senddata.config(font="bold")

        self.button\_senddata.grid(column=5, row=1)

        #Refresh Rate entry box

        self.senddata\_edit = tkinter.Entry(window,width=8)

        self.senddata\_edit.grid(column=4,row=1)

        self.senddata\_edit.config(font="bold")

        self.senddata\_edit.insert(tkinter.END,"10")

        self.refresh\_r=10

        # Textbox

        self.textbox=tkscrolledtext.ScrolledText(master=window, wrap='word', width=50, height=15) #width=characters, height=lines

        self.textbox.grid(column=1, row=3,columnspan=5)

        self.textbox.config(font="bold")

        # Button that lets the user take a snapshot

        self.btn\_snapshot=tkinter.Button(window, text="Snapshot", width=50, command=self.snapshot)

        self.btn\_snapshot.grid(column=0,row=10)

        ##### start multithread

        self.new\_thread1 = threading.Timer(self.refresh\_r,

            function=self.pic\_to\_text,

            kwargs={})

        self.new\_thread1.daemon=True

        self.new\_thread1.start()

        self.window.after(200, self.update\_Image)

############### Take the image and convert it to Text and Send

    def pic\_to\_text(self):

        #Add new thread to multithread

        self.new\_thread1 = threading.Timer(self.refresh\_r,

            function=self.pic\_to\_text,

            kwargs={})

        self.new\_thread1.daemon=True

        self.new\_thread1.start()

        ##### if com port is open

        if serialPort.IsOpen():

            time\_now=str(datetime.datetime.now()) # get time now

            ## convert image to tex

            text1=Img\_to\_Word.main(self.photo\_img)

            message=time\_now + " " + text1

            message += '\r\n'

            print(message)

            serialPort.Send(message)

            self.textbox.insert('1.0',message)

        else: ## if comport is closed

            self.textbox.insert('1.0', "Not sent - COM port is closed\r\n")

    ### used for snapshot button to extract text from image at a single moment

    def snapshot(self):

        time\_now=str(datetime.datetime.now())

        #### image to text

        text1=Img\_to\_Word.main(self.photo\_img)

        message=time\_now+ " " + text1 #+'\r\n'

        ### used to save images

        #img =cv2.cvtColor(self.photo\_img,cv2.COLOR\_RGB2GRAY)

        #ne=time\_now.split(".")

        #wr=ne[1]+".jpg"

        #cv2.imwrite(wr,img)

        message += '\r\n'

        print(message)

        if serialPort.IsOpen(): # if comport is open send message

            serialPort.Send(message)

            self.textbox.insert('1.0',message)

        else:# closed comport: write message but will not send

            self.textbox.insert('1.0',("WARNING closed comport, message not sent:   " + message))

    # Used to udate video in Tinker

    def update\_Image(self):

        # Get a frame from the video source

        ret, frame = self.vid.get\_frame()

        ### Get image

        if ret:

            self.photo\_img=frame

            self.photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))

            self.canvas.create\_image(0, 0, image = self.photo, anchor = tkinter.NW)

        #### Add thread to video queue

        try:

            res = self.thread\_queue.get(0)

            self.text\_label.config(text=res)

            self.window.after(100, self.update\_Image)

        except queue.Empty:

            self.window.after(100, self.update\_Image)

    ### clear the text box

    def ClearDataCommand(self):

        self.textbox.delete("1.0",tkinter.END)

    # # serial data callback function

    def OnReceiveSerialData(self,message):

        str\_message = message.decode("utf-8")

        self.textbox.insert('1.0', str\_message)

    # Register the callback above with the serial port object

    serialPort.RegisterReceiveCallback(OnReceiveSerialData)

    ## open the serial comport

    def OpenCommand(self):

        if self.button\_openclose.cget("text") == 'Open COM Port':

            comport = self.comport\_edit.get()

            baudrate = self.baudrate\_edit.get()

            serialPort.Open(comport,baudrate)

            self.button\_openclose.config(text='Close COM Port')

            self.textbox.insert('1.0', "COM Port Opened\r\n")

        elif self.button\_openclose.cget("text") == 'Close COM Port':

            serialPort.Close()

            self.button\_openclose.config(text='Open COM Port')

            self.textbox.insert('1.0',"COM Port Closed\r\n")

    ######## Change the refresh rate

    def Refresh\_Rate(self):

        plholder=self.senddata\_edit.get()

        #### tends to crash if refresh rate is to fast

        if int(plholder)>4.99:

            self.textbox.insert('1.0',("Refresh rate changed: "+plholder+ "\r\n"))

            self.refresh\_r = int(plholder)

        else:

            self.textbox.insert('1.0',("Refresh rate WAS NOT changed please choose rate greater then 5s. \r\n"))

####### Create the video capture

class MyVideoCapture:

    def \_\_init\_\_(self, video\_source=0):

        # Open the video source

        self.vid = cv2.VideoCapture(video\_source)

        if not self.vid.isOpened():

            raise ValueError("Unable to open video source", video\_source)

        # Get video source width and height

        self.width = self.vid.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

        self.height = self.vid.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

    def get\_frame(self):

        if self.vid.isOpened():

            ret, frame = self.vid.read()

            if ret:

                # Return a boolean success flag and the current frame converted to BGR

                return (ret, cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB))

            else:

                return (ret, None)

        else:

            return (None, None)

    # Release the video source when the object is destroyed

    def \_\_del\_\_(self):

        if self.vid.isOpened():

            self.vid.release()

#### multi threading

class ThreadedTask(threading.Thread):

    def \_\_init\_\_(self, queue):

        threading.Thread.\_\_init\_\_(self)

        self.thread\_queue = queue

    def run(self):

        time.sleep(5)  # Simulate long running process

        self.thread\_queue.put("Task finished")

### initialize tinker

root=tkinter.Tk()

App(root, "Tkinter and OpenCV")

### run system

root.mainloop()

sys.exit()

root.destroy()

Appendix A.2. Img\_to\_Word.py

import cv2

import numpy as np

import pytesseract

import imutils

import re

##### From the given a word and a number find the dose information

# because of the OCR nature we have two list one number (is just the tessaract OCR system looking only for numbers)

# word (is tessaract OCR looking for numbers and letters)

def order\_output(word,number):

    ### create a list from all words and numbers

    words=word.upper().split("\n")

    numbers=number.split("\n")

    #print(words)

    #print(numbers)

    output="error" # final output of the data

    agent="" # agent name used

    regex = r'[0-9]' # sort for numbers

    numberlist=[] #list of numbers

    wordlist=[] #list of words

    nprev="" #placeholder previous nuber found

    nextnu=False # next string is a number?

    #### Go through the numbers for dose amount

    for n in numbers:

        if (re.match("\_", n)!=None):#Blank found

            nprev=None

            continue

        if (re.match(regex, n)==None) or (len(n)<1):#no numbers found

            continue

        if nextnu==True:# string is a number

            numberlist.append(n+" " + nprev)#append it to the number list

            nextnu=False

        if ":" in n:# is it two number seprated by a :

            if n.rfind(":")<4:

                if nprev!=None:

                    numberlist.append(nprev+" " + n)

                else:

                    nextnu=True

        nprev=n

    numb2=[]

    pastele=''

    ###### go through the list of words

    for ele in words:

        if len(ele)<1:#is the word a blank ie. <1

            pastele=ele

            continue

        elif ele in pastele == True: # OCR will some times treat non-letter as letters they often are repeated

            pastele=ele

            continue

        pastele=ele

        if re.findall("[0-9]",ele)!=[]:# does the element have a number

            if re.findall("[a-zA-Z]",ele)!=[]: #does it have a letter

                non=1

            else: # the element maybe a number just not found

                numb2.append(ele)

        # dose thee element slightly match the desired values

        if "FENT" in ele or "TENT" in ele or "NTAN" in ele: # does the element match one of these then we assume

            if agent != "fentaNYL 1000: ": # is the past agent being repeated

                agent="fentaNYL 1000: "

                wordlist.append(agent)#add to the agent list

                numb2.append("dose\_here")#add the dose here

        elif "SODI" in ele or "SODJ" in ele or "SADI" in ele:

            if agent != "Sodium Cl 3%: ":

                agent="Sodium Cl 3%: "

                wordlist.append(agent)

                numb2.append("dose\_here")

    #print(words)

    #print(wordlist)

    #print(numb2)

    #print(numberlist)

    ### compare the list of words and number to create the dosing information

    if len(numberlist)==len(wordlist):# if they are the same then we have only one dose amount and agent

        output=""

        for i in range(len(numberlist)):

            output+=(wordlist[i]+" "+numberlist[i]+" ")

        return output

    elif numb2.count("dose\_here")==len(wordlist):#if we have a miss match replace dose here with actual amount

        stto=0

        output=""

        for i in range(len(wordlist)):

            stto= numb2.index("dose\_here",stto)+1

            if numb2[stto-3] != "dose\_here":# is volume recored or not

                output+=(wordlist[i]+" "+numb2[stto-3]+" ")

            else:

                output+=(wordlist[i]+" "+numb2[stto-2]+" ")

        return output

    else:# there is a missmatch between dose agent and amount

        output= ["Error miss match"]

        return output

#### resize the image

def resize\_img(img, percent):

    width = int(img.shape[1] \* percent / 100)

    height = int(img.shape[0] \* percent / 100)

    dim = (width, height)

    # resize image

    imgout = cv2.resize(img, dim, interpolation = cv2.INTER\_CUBIC)

    return imgout

### used to find borders of key image

def find\_border\_components(contours, ary):

    key\_contours=[]

    img\_h=ary.shape[0]

    img\_w=ary.shape[1]

    for i, c in enumerate(contours):

        x,y,w,h = cv2.boundingRect(c)

        if (h\*w)>(0.3\*img\_h\*img\_w):

            key\_contours.append((x,y,w,h))

    if len(key\_contours)==0:

        return

    key\_contours.sort(key=lambda x:(x[2]\*x[3]))

    #print(key\_contours)

    borders=(key\_contours[0][0],key\_contours[0][1],key\_contours[0][0]+key\_contours[0][2]+key\_contours[0][0],key\_contours[0][1]+key\_contours[0][3])

    return borders

#### main function takes an image and converts it to dosing information

def main(imgin):#input is a RGB array image

    img =cv2.cvtColor(imgin,cv2.COLOR\_RGB2GRAY)# convert to greyscale

    #img=cv2.imread(imgin,0)

    edges = cv2.Canny(img,100,200) # find all the edges

    kernel = np.ones((2,2), dtype=np.uint8) # define a kernel size used to find the edges

    edges=cv2.dilate(edges, kernel, iterations=3) # find the images edges

    #cv2.imwrite("B.jpg",edges)

    contours, hierarchy = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE) #use edges to create contours

    borders = find\_border\_components(contours, edges) # Note the main disply must be atleast 50% of display window

    if borders!=None:

        img=img[borders[1]:borders[3],borders[0]:borders[2]]

    ### resize the image based on borders

    img = resize\_img(img,400)

    #cv2.imwrite("C.jpg",img)

    ### black and white the image to get solid letters and numbers

    if np.median(img) < 200:

        img = 255-img

    img=cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY, 101, 23)

    #cv2.imwrite("D.jpg",img)

    ## Again find the edges and then reshape

    edges = cv2.Canny(img,100,200)

    kernel = np.ones((3,3), dtype=np.uint8)

    edges=cv2.dilate(edges, kernel, iterations=6)

    #cv2.imwrite("E.jpg",edges)

    #### check for image rotation

    contours, hierarchy = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    height, width = img.shape

    h\_len=height

    rect=((width/2,height/2),(height,width),0)

    for i, c in enumerate(contours):

        ((x,y), (h,w),rotation) = cv2.minAreaRect(c)

        if w>width\*0.5:

            if h<h\_len:

                rect=cv2.minAreaRect(c)

                h\_len=h

    ((x,y), (h,w),rotation)=rect

    box = cv2.boxPoints(rect)

    box = np.int0(box)

    img2=cv2.cvtColor(img,cv2.COLOR\_GRAY2RGB)

    img2=cv2.drawContours(img2,[box],0,(0,0,255),10)

    #print(rotation)

    ####  rotate image

    if rotation!=None and rotation<-45:

        img = imutils.rotate(img, angle=90+rotation)

    #### trim after rotation

    if borders!=None:

        if round(x-w/2)<60:

            img=img[round(height\*0.05):round(height\*0.9),60:round(x+w/2)]

        else:

            img=img[round(height\*0.05):round(height\*0.9),round(x-w/2):round(x+w/2)]

    #cv2.imwrite("F.jpg",img)

    ## use tesseract to assess the image

    d = pytesseract.image\_to\_data(img, output\_type=pytesseract.Output.DICT)

    # from the tesseract image found object extract them and put them into a vertical image

    n\_boxes = len(d['level'])

    maxw=np.max(d['width'])

    full=np.ones((1,round(maxw/2)+10))\*255

    for i in range(n\_boxes):

        (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])

        #cv2.rectangle(img4, (x, y), (x + w, y + h), (0, 255, 0), 2)

        if w<round(maxw/2):

            if h<200:

                bas=np.ones((h+20,round(maxw/2)+10))\*255

                #print(h,w,maxw,x+w-x,w+10-10,round(maxw/2))

                crop\_img = img[y:(y+h), x:(x+w)]

                bas[0:h, 10:(w+10)]=crop\_img

                full=np.vstack((full,bas))

    img=full

    #cv2.imwrite("G.jpg",img)

    #### use OCR tesseract to find all words and numbers

    custom\_config = r'-c tessedit\_char\_whitelist=0123456789.:\_ --psm 11'

    final\_num= (pytesseract.image\_to\_string(img, config=custom\_config))

    custom\_config = r'-c tessedit\_char\_blacklist=\"--psm 11'

    final\_text= (pytesseract.image\_to\_string(img, config=custom\_config))

    ###### sort the words and numbers from OCR

    worder= order\_output(final\_text,final\_num)

    #print(worder)

    return worder

Appendix A.3. serial\_rx\_tx.py

import serial

import sys

import \_thread

#### declaring a serial port

class SerialPort:

    def \_\_init\_\_(self):

        self.comportName = ""

        self.baud = 0

        self.timeout = None

        self.ReceiveCallback = None

        self.isopen = False

        self.receivedMessage = None

        self.serialport = serial.Serial()

    def \_\_del\_\_(self):

        try:

            if self.serialport.is\_open():

                self.serialport.close()

        except:

            print("Error, closing COM port: ", sys.exc\_info()[0] )

    def RegisterReceiveCallback(self,aReceiveCallback):

        self.ReceiveCallback = aReceiveCallback

        try:

            \_thread.start\_new\_thread(self.SerialReadlineThread, ())

        except:

            print("Error, starting thread read: ", sys.exc\_info()[0])

    def SerialReadlineThread(self):

        while True:

            try:

                if self.isopen:

                    self.receivedMessage = self.serialport.readline()

                    if self.receivedMessage != "":

                        self.ReceiveCallback(self.receivedMessage)

            except:

                print("Error reading COM port: ", sys.exc\_info()[0])

    def IsOpen(self):

        return self.isopen

    def Open(self,portname,baudrate):

        if not self.isopen:

            self.serialport.port = portname

            self.serialport.baudrate = baudrate

            try:

                self.serialport.open()

                self.isopen = True

            except:

                print("Error, opening COM port: ", sys.exc\_info()[0])

    def Close(self):

        if self.isopen:

            try:

                self.serialport.close()

                self.isopen = False

            except:

                print("Close error, closing COM port: ", sys.exc\_info()[0])

    def Send(self,message):

        if self.isopen:

            try:

                # Ensure that the end of the message has both \r and \n, not just one or the other

                newmessage = message.strip()

                newmessage += '\r\n'

                self.serialport.write(newmessage.encode('utf-8'))

            except:

                print("Error, sending message: ", sys.exc\_info()[0] )

            else:

                return True

        else:

            return False