Contents

[Servomotor 2](#_Toc116570132)

[servo.py 3](#_Toc116570133)

[IMU 5](#_Toc116570134)

[MPU\_Measure.py 6](#_Toc116570135)

[Load Cell 10](#_Toc116570136)

[load\_cell.py 11](#_Toc116570137)

[Camera 14](#_Toc116570138)

[usb\_camera.py 15](#_Toc116570139)

[pi\_camera\_image.py 16](#_Toc116570140)

[pi\_camera\_video.py 17](#_Toc116570141)

[Image Analysis 18](#_Toc116570142)

[process\_img.py 19](#_Toc116570143)

[toBinary.py 23](#_Toc116570144)

[rescale\_frame.py 24](#_Toc116570145)

[getEdges.py 25](#_Toc116570146)

[getBoundingBox.py 26](#_Toc116570147)

[getFoamFromModel.py 27](#_Toc116570148)

[getDigestateInfo.py 29](#_Toc116570149)

[colourBlockDetection.py 31](#_Toc116570150)

[client\_pi.py 33](#_Toc116570151)

[server\_host.py 35](#_Toc116570152)

[maskRCNN\_training.py 37](#_Toc116570153)

[Remote Control Web-app 40](#_Toc116570154)

[index.php 41](#_Toc116570155)

[main.php 42](#_Toc116570156)

[servo.php 46](#_Toc116570157)

[foam.php 51](#_Toc116570158)

[main.css 56](#_Toc116570159)

[Servo[0-180].py 59](#_Toc116570160)

# Servomotor

## servo.py

Author(s): Jason Ltg

from gpiozero import Servo

import math

from time import sleep

from gpiozero.pins.pigpio import PiGPIOFactory

factory = PiGPIOFactory()

servo = Servo(17, min\_pulse\_width = 0.5/1000, max\_pulse\_width = 2.33/1000, pin\_factory=factory)

servo.min()

try:

    while True:

        #Ask user for angle and turn servo to it

        try:

            angle = float(input('Enter an angle between 0 & 180: '))

        except ValueError:

            print("No... Input is not a number")

            continue

        else:

            if angle > 180:

                print('Not so high, between 0 and 180 please!')

                continue

            elif angle < 0:

                print('No negatives, between 0 and 180 please!')

                continue

            # print("Type of number ", type(angle))

            servo.value = math.cos(math.radians(angle)) \* -1

            sleep(1)

except KeyboardInterrupt:

    sleep(1)

    servo.mid()

    sleep(1)

    #servo.stop() // These two lines are not functions of the library...

    #GPIO.cleanup()

    print("\nGoodbye!")

#while True:

#    for i in range (0, 360):

#        print(i)

#        servo.value = math.sin(math.radians(i))

#        sleep(0.01)

#while True:

 #   for i in range (0, 360,10):

  #      print(i)

   #     servo.value = math.sin(math.radians(i))

    #    sleep(0.1)

#servo.min()

#sleep(1)

#servo.mid()

#sleep(1)

#servo.max()

#sleep(1)

#servo.value = -1

#sleep(1)

#servo.value = 0

#sleep(1)

#servo.value = 1

#sleep(1)

# IMU

## MPU\_Measure.py

Author(s): John Nguyen

import smbus #import SMBus module of I2C

from time import sleep          #import

import math

import time

import cv2

#some MPU6050 Registers and their Address

PWR\_MGMT\_1   = 0x6B

SMPLRT\_DIV   = 0x19

CONFIG       = 0x1A

GYRO\_CONFIG  = 0x1B

INT\_ENABLE   = 0x38

ACCEL\_XOUT\_H = 0x3B

ACCEL\_YOUT\_H = 0x3D

ACCEL\_ZOUT\_H = 0x3F

GYRO\_XOUT\_H  = 0x43

GYRO\_YOUT\_H  = 0x45

GYRO\_ZOUT\_H  = 0x47

def MPU\_Init():

    #write to sample rate register

    bus.write\_byte\_data(Device\_Address, SMPLRT\_DIV, 7)

    #Write to power management register

    bus.write\_byte\_data(Device\_Address, PWR\_MGMT\_1, 1)

    #Write to Configuration register

    bus.write\_byte\_data(Device\_Address, CONFIG, 0)

    #Write to Gyro configuration register

    bus.write\_byte\_data(Device\_Address, GYRO\_CONFIG, 24)

    #Write to interrupt enable register

    bus.write\_byte\_data(Device\_Address, INT\_ENABLE, 1)

def read\_raw\_data(addr):

    #Accelero and Gyro value are 16-bit

        high = bus.read\_byte\_data(Device\_Address, addr)

        low = bus.read\_byte\_data(Device\_Address, addr+1)

        #concatenate higher and lower value

        value = ((high << 8) | low)

        #to get signed value from mpu6050

        if(value > 32768):

                value = value - 65536

        return value

bus = smbus.SMBus(1) # or bus = smbus.SMBus(0) for older version boards

Device\_Address = 0x68   # MPU6050 device address

MPU\_Init()

# print (" Reading Data of Gyroscope and Accelerometer")

def Average(l):

    avg = sum(l) / len(l)

    return avg

t = 0

angle\_x = 0

angle\_y = 0

angle\_z = 0

pi = math.pi

avg\_Ax = [0]\*20

avg\_Ay = [0]\*20

avg\_single\_Ax = 0

avg\_single\_Ay = 0

Vx = 0

Vy = 0

Vz = 0

prev\_time = time.time()

Ax\_max = 0

Ay\_max = 0

Az\_max = 0

while cv2.waitKey(100) != "q":

    # for i in range(20):

        #Read Accelerometer raw value

    acc\_x = read\_raw\_data(ACCEL\_XOUT\_H)

    acc\_y = read\_raw\_data(ACCEL\_YOUT\_H)

    acc\_z = read\_raw\_data(ACCEL\_ZOUT\_H)

        #Read Gyroscope raw value

    gyro\_x = read\_raw\_data(GYRO\_XOUT\_H)

    gyro\_y = read\_raw\_data(GYRO\_YOUT\_H)

    gyro\_z = read\_raw\_data(GYRO\_ZOUT\_H)

        #Full scale range +/- 250 degree/C as per sensitivity scale factor

    Ax = -0.02 + acc\_x/16384.0

    Ay = -0.07 + acc\_y/16384.0

    Az = 1.06 + acc\_z/16384.0

    if abs(Ax) > abs(Ax\_max):

        Ax\_max = Ax

    if abs(Ay) > abs(Ay\_max):

        Ay\_max = Ay

    if abs(Az) > abs(Az\_max):

        Az\_max = Az

    Gx = 0.56 + gyro\_x/131.0

    Gy = 0.19 + gyro\_y/131.0

    Gz = -0.19 + gyro\_z/131.0

        #avg\_Ax[i] = Ax

        #avg\_Ay[i] = Ay

    #avg\_single\_Ax = Average(avg\_Ax);

    #avg\_single\_Ay = Average(avg\_Ay);

    #angle\_x = round(avg\_single\_Ax \* 90, 4) # @90d Ax = 1... 1\*x=90, x=90

    #angle\_y = round(avg\_single\_Ay \* 91.4, 4) # @90d Ay=0.985... 0.985\*y=90, y=91.4

    dt = time.time()-prev\_time

    prev\_time = time.time()

    # if abs(Ax) <0.05:

       # Ax = 0

    # if abs(Ay) <0.05:

      #   Ay = 0

    # if abs(Ay) <0.05:

      #   Ay = 0

    Vx += Ax \*dt

    Vy += Ay \*dt

    Vz += Az \*dt

    # print ("Gx=%.2f" %Gx, u'\u00b0'+ "/s", "\tGy=%.2f" %Gy, u'\u00b0'+ "/s", "\tGz=%.2f" %Gz, u'\u00b0'+ "/s", "\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)

    #sleep(0.1)

    # print ("\tTime=%.2f s" %t)

    # print("acc\_x:", Ax, "acc\_y:", Ay, "acc\_z:", Az)

    # Sprint("angle\_x:", angle\_x, "angle\_y:", angle\_y)

    #print(str(round(angle\_y)))

    #print(str(round(angle\_x)))

    print ("Current Gx=%.2f" %Gx, "\tCurrent Gy=%.2f" %Gy,"\tCurrent Gz=%.2f" %Gz)

    #print("avg\_Ax=%.2f" %avg\_single\_Ax, "avg\_Ay=%.2f" %avg\_single\_Ay)

    #print ("angle\_x=%.2f" %angle\_x, "\tangle\_y=%.2f" %angle\_y)

    print ("Current Ax=%.2f" %Ax, "\tCurrent Ay=%.2f" %Ay,"\tCurrent Az=%.2f" %Az)

    print ("Max Ax=%.2f" %Ax\_max, "\t\tMax Ay=%.2f" %Ay\_max,"\t\tMax Az=%.2f" %Az\_max)

    # print("Vx=%.2f" %Vx, "\tVy=%.2f" %Vy,"\tVz=%.2f" %Vz)

    # print ("gyro\_x=%.2f" %gyro\_x, "\tgyro\_y=%.2f" %gyro\_y,"\tgyro\_z=%.2f" %gyro\_z)

    # print ("acc\_x=%.2f" %acc\_x, "\tacc\_y=%.2f" %acc\_y,"\tacc\_z=%.2f" %Gz)

# Load Cell

## load\_cell.py

Author(s): John Nguyen

#! /usr/bin/python2

import time

import sys

import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

EMULATE\_HX711=False

referenceUnit = 201

offset = 5

if not EMULATE\_HX711:

    import RPi.GPIO as GPIO

    from hx711 import HX711

else:

    from emulated\_hx711 import HX711

def cleanAndExit():

    print("Cleaning...")

    if not EMULATE\_HX711:

        GPIO.cleanup()

    print("Bye!")

    sys.exit()

def button\_callback(channel):

    offset = val

    print('offset when tare =',round(offset,3))

    print("Tare done! Add weight now...")

    hx.reset()

    hx.tare()

    time.sleep(1)

GPIO.setwarnings(False) # Ignore warning for now

GPIO.setmode(GPIO.BCM) # Use physical pin numbering

GPIO.setup(16, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN) # Set pin 10 to be an input pin and set initial value to be pulled low (off)

GPIO.add\_event\_detect(16,GPIO.RISING,callback=button\_callback) # Setup event on pin 10 rising edge

hx = HX711(5, 6)

# I've found out that, for some reason, the order of the bytes is not always the same between versions of python, numpy and the hx711 itself.

# Still need to figure out why does it change.

# If you're experiencing super random values, change these values to MSB or LSB until to get more stable values.

# There is some code below to debug and log the order of the bits and the bytes.

# The first parameter is the order in which the bytes are used to build the "long" value.

# The second paramter is the order of the bits inside each byte.

# According to the HX711 Datasheet, the second parameter is MSB so you shouldn't need to modify it.

hx.set\_reading\_format("MSB", "MSB")

# HOW TO CALCULATE THE REFFERENCE UNIT

# To set the reference unit to 1. Put 1kg on your sensor or anything you have and know exactly how much it weights.

# In this case, 92 is 1 gram because, with 1 as a reference unit I got numbers near 0 without any weight

# and I got numbers around 184000 when I added 2kg. So, according to the rule of thirds:

# If 2000 grams is 184000 then 1000 grams is 184000 / 2000 = 92.

#hx.set\_reference\_unit(113)

hx.set\_reference\_unit(referenceUnit)

hx.reset()

hx.tare()

# print("Tare done! Add weight now...")

# to use both channels, you'll need to tare them both

#hx.tare\_A()

#hx.tare\_B()

while True:

    try:

        # These three lines are usefull to debug wether to use MSB or LSB in the reading formats

        # for the first parameter of "hx.set\_reading\_format("LSB", "MSB")".

        # Comment the two lines "val = hx.get\_weight(5)" and "print val" and uncomment these three lines to see what it prints.

        # np\_arr8\_string = hx.get\_np\_arr8\_string()

        # binary\_string = hx.get\_binary\_string()

        # print binary\_string + " " + np\_arr8\_string

        # Prints the weight. Comment if you're debbuging the MSB and LSB issue.

        val = hx.get\_weight(5)

        #if GPIO.input(16) == 1 and val<2000:

        #   offset = val

        if val > 2000:

            val = 0

        print(str(round(val,3)))

        # Open a file with access mode 'a'

        file\_object = open('weights.txt', 'a')

        # Append 'hello' at the end of file

        file\_object.write(str(round(val,3))+"\n")

        # Close the file

        file\_object.close()

        # To get weight from both channels (if you have load cells hooked up

        # to both channel A and B), do something like this

        #val\_A = hx.get\_weight\_A(5)

        #val\_B = hx.get\_weight\_B(5)

        #print "A: %s  B: %s" % ( val\_A, val\_B )

        hx.power\_down()

        hx.power\_up()

        time.sleep(0.01)

    except (KeyboardInterrupt, SystemExit):

        cleanAndExit()

# Camera

## usb\_camera.py

Author(s): Ling Chen

import cv2

camera = cv2.VideoCapture(0)

# print(camera.isOpened())

ret, image = camera.read()

cv2.imwrite("foam\_img.jpg", image)

camera.release()

del(camera)

## pi\_camera\_image.py

Author(s): Ritchie Leong

from picamera import PiCamera

import time

camera = PiCamera()

camera.resolution = (640, 960) #set resolution

camera.vflip = False #set the camera upright

# camera.start\_preview() #camera display on the screen

time.sleep(2) #sleep 5 seconds

#take photo after running the code for 2 seconds

camera.capture("foam\_img.png")

#image taken is saved as jpg file in the same directory. Saved on desktop in this case

camera.close()

# time.sleep(1) #sleep 1 seconds

#  camera.stop\_preview() #close camera display on the screen

## pi\_camera\_video.py

Author(s): Ritchie Leong

from picamera import PiCamera

import time

camera = PiCamera()

#set camera resolution

camera.resolution = (640, 360)

#set camera position upright

camera.vflip = True

#turn on camera after 5 seconds when code runs

camera.start\_preview()

time.sleep(5)

#start recording and stop after 3 seconds

camera.start\_recording("videotest.h264") #video is saved as .h264 file

time.sleep(3)

camera.stop\_recording()

time.sleep(2) #sleep 2 seconds

camera.stop\_preview() #close camera display on the screen

# Image Analysis

## process\_img.py

Author(s): Ling Chen

import numpy as np

import sys

sys.path.append("foam\_detection/")

sys.path.append("foam\_detection/detectron2-main/")

sys.path.append("detectron2-main/")

sys.path.append("foam\_detection/rembg-main/")

from utils.rescaleFrame import rescaleFrame

from utils.getEdges import getEdges

from utils.getDigestateInfo import getDigestateInfo

from utils.toBinary import toBinary

from utils.getBoundingBox import getBoundingBox

from utils.getFoamFromModel import getFoamFromModel

from utils.colourBlockDetection import colourBlockDetection

import rembg

import cv2

def process\_img(image=None):

    # image = cv2.imread("foam\_detection/images/clear.jpg")

    # image = cv2.imread("foam\_detection/images/fg1.jpg")

    if image is None:

        image = cv2.imread("foam\_detection/foaming\_images/test2.jpg")

    # image = cv2.imread("foam\_detection/foaming\_images/Yellowx3\_foamx5.jpg")

    scale = 720/max(image.shape[0],image.shape[1])

    image = rescaleFrame(image, scale)

    # cv2.imshow("original\_image", image)

    #########################################################################################################

    # Removing the backgorund using AI method from rembg package

    flask\_img = rembg.remove(image)

    flask\_img = cv2.cvtColor(flask\_img, cv2.COLOR\_BGRA2BGR)

    # # Optional: Remove Glare

    # flask\_img = removeGlare(flask\_img, 210)

    cv2.imshow("removed\_background", flask\_img)

    # foam - foam height - digestate - digestate height

    viz0000 = flask\_img.copy()

    viz0001 = flask\_img.copy()

    viz0010 = flask\_img.copy()

    viz0011 = flask\_img.copy()

    viz0100 = flask\_img.copy()

    viz0101 = flask\_img.copy()

    viz0110 = flask\_img.copy()

    viz0111 = flask\_img.copy()

    viz1000 = flask\_img.copy()

    viz1001 = flask\_img.copy()

    viz1010 = flask\_img.copy()

    viz1011 = flask\_img.copy()

    viz1100 = flask\_img.copy()

    viz1101 = flask\_img.copy()

    viz1110 = flask\_img.copy()

    viz1111 = flask\_img.copy()

    viz\_list = [viz0000, viz0001, viz0010, viz0011, viz0100, viz0101, viz0110, viz0111, viz1000, viz1001, viz1010, viz1011, viz1100, viz1101, viz1110, viz1111]

    viz\_list\_str = ["viz0000", "viz0001", "viz0010", "viz0011", "viz0100", "viz0101", "viz0110", "viz0111", "viz1000", "viz1001", "viz1010", "viz1011", "viz1100", "viz1101", "viz1110", "viz1111"]

    #########################################################################################################

    # Get the bounding box of the flask

    box = getBoundingBox(flask\_img)

    x, y, w, h = box

    #########################################################################################################

    # Get bounding box of the foam using trained mask rcnn model

    no\_foam = False

    n = 0

    for i in range(len(viz\_list)):

        if viz\_list\_str[i][-4] == "1":

            if n == 0:

                n+=1

                foam\_bbox, viz\_list[i] = getFoamFromModel(viz\_list[i])

                foam\_viz = viz\_list[i].copy()

            else:

                viz\_list[i] = foam\_viz.copy()

    if foam\_bbox == None:

        no\_foam = True

    else:

        foam\_edge = round(max(foam\_bbox[3], foam\_bbox[1]))

    # cv2.imshow("foam\_viz", viz\_list[-1])

    #########################################################################################################

    # Extract important digestate info from image and canny edges

    digestate\_info = getDigestateInfo(flask\_img, box, foam\_bbox, no\_foam)

    liquid\_colour = digestate\_info["digestate colour"]

    liquid\_height = digestate\_info["digestate height"]

    true\_liquid\_height = digestate\_info["real digestate height"]

    if not no\_foam:

        foam\_height = digestate\_info["foam height"]

        foam\_colour = digestate\_info["foam colour"]

        true\_foam\_height = digestate\_info["real foam height"]

    #########################################################################################################

    # Get bounding box of colours

    for i in range(len(viz\_list)):

        if viz\_list\_str[i][-2] == "1":

            viz\_list[i] = colourBlockDetection(flask\_img, viz\_list[i], liquid\_colour)

    #########################################################################################################

    # Displaying the detected colour

    colour = np.zeros((100,100,3), dtype = "uint8")

    for i in range(100):

        for j in range(100):

            colour[i][j] = liquid\_colour

    # cv2.imshow("colour\_of\_liquid", colour)

    # Displaying the detected foam colour

    if not no\_foam:

        colour\_foam = np.zeros((100,100,3), dtype = "uint8")

        for i in range(100):

            for j in range(100):

                colour\_foam[i][j] = foam\_colour

        # cv2.imshow("colour\_of\_foam", colour\_foam)

    #########################################################################################################

    # Displaying height lines

    if no\_foam:

        for i in range(len(viz\_list)):

            if viz\_list\_str[i][-1] == "1":

                # Displaying the detected digestate/liquid height

                cv2.line(viz\_list[i], (x+(w//2), y+h-(liquid\_height)), (x+(w//2),y+h), (0, 0, 255), 2)

                string = "digestate height: " + str(round(true\_liquid\_height\*1000,2))  +  " mm"

                cv2.putText(viz\_list[i],string, (x, min(y+h +50, viz\_list[i].shape[0]-25)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255), 1)

                # print(string)

    else:

        for i in range(len(viz\_list)):

            if viz\_list\_str[i][-1] == "1":

                # Displaying the detected digestate/liquid height

                cv2.line(viz\_list[i], (x+(w//2), foam\_edge), (x+(w//2),foam\_edge+(liquid\_height)), (0, 0, 255), 2)

                string = "digestate height: " + str(round(true\_liquid\_height\*1000,2))  + " mm"

                cv2.putText(viz\_list[i],string, (x, min(y+h +50, viz\_list[i].shape[0]-25)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255), 1)

                # print(string)

        for i in range(len(viz\_list)):

            if viz\_list\_str[i][-3] == "1":

                # Displaying the detected foam height

                cv2.line(viz\_list[i], (x+(w//2), round(foam\_bbox[1])), (x+(w//2), round(foam\_bbox[3])), (0, 255, 0), 2)

                string = "foam height: " + str(round(true\_foam\_height\*1000,2))  + " mm"

                cv2.putText(viz\_list[i],string, (x, max(round(min(foam\_bbox[1], foam\_bbox[3]) -50), 25)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,255,0), 1)

                # print(string)

            # cv2.imshow("final\_output", viz)

    # cv2.waitKey(0)

    # cv2.destroyAllWindows()

    return viz\_list, viz\_list\_str, colour, colour\_foam, digestate\_info

if \_\_name\_\_ == "\_\_main\_\_":

    process\_img()

## toBinary.py

Author(s): Ling Chen

import cv2

def toBinary(image):

    """

    Function to convert an image to binary version using OTSU thresholding

    :param image: The OpenCV cv::Mat BGR image

    :returns: Binary image

    """

    # Convert image to grayscale

    gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    # Apply Bilateral Blurring (to reduce noise while keeping edges sharp)

    blur = cv2.bilateralFilter(gray, 5, 75, 75)

    # Converting blurred image into binary image

    thresh = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY\_INV)[1]

    # cv2.imshow('thresh',thresh)

    return thresh

## rescale\_frame.py

Author(s): Ling Chen

import cv2

# Rescaling image

def rescaleFrame(frame, scale):

    """

    Function to rescale an image based on a scale ratio

    :param frame: image to rescale

    :param scale: positive float representing scale

    :returns: rescaled image

    """

    width = int(frame.shape[1] \* scale)

    height = int(frame.shape[0] \* scale)

    dimensions = (width, height)

    return cv2.resize(frame, dimensions, interpolation=cv2.INTER\_CUBIC)

## getEdges.py

Author(s): Ling Chen

import cv2

import numpy as np

def getEdges(image, blur):

    """

    function to get the canny edges of an image after applying shapening and blurring.

    :param image: image to find the edges

    :param blur: odd number for the blur kernel

    :returns: the edges from the image

    """

    if blur%2 == 0:

        raise Exception("Blur must be odd integer")

    kernel = np.array([[0, -1, 0],

                    [-1, 5,-1],

                    [0, -1, 0]], )

    image\_sharp = cv2.filter2D(src=image, ddepth=-1, kernel=kernel)

    # cv2.imshow('sharpened image', image\_sharp)

    # Apply Bilateral Blurring (to reduce noise while keeping edges sharp)

    # blurred = cv2.bilateralFilter(image\_sharp, blur, 75, 75)

    blurred = cv2.GaussianBlur(image\_sharp,(blur,blur),0)

    canny\_edges = cv2.Canny(blurred, 45, 155)

    return canny\_edges

## getBoundingBox.py

Author(s): Ling Chen

import cv2

from utils.getEdges import getEdges

from utils.toBinary import toBinary

def getBoundingBox(image):

    binary = toBinary(image)

    canny = getEdges(binary, 1)

    # cv2.imshow("can", canny)

    # cv2.imshow("bi",binary)

    # cv2.imshow("ca",canny)

    # cv2.waitKey(0)

    contours, hierarchy = cv2.findContours(canny, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

    contours = sorted(contours, key=cv2.contourArea, reverse=True)

    best\_contour = contours[0]

    box = cv2.boundingRect(best\_contour)

    return box

## getFoamFromModel.py

Author(s): Ling Chen

import sys

sys.path.append("./foam\_detection/detectron2-main/")

sys.path.append("./detectron2-main/")

import os

import pickle

from detectron2.engine import DefaultPredictor

from detectron2.utils.visualizer import Visualizer

from detectron2.utils.visualizer import ColorMode

import cv2

def getFoamFromModel(image):

    cfg\_save\_path = "foam\_mask\_rcnn/IS\_cfg.pickle"

    with open(cfg\_save\_path, 'rb') as f:

        cfg = pickle.load(f)

    cfg.MODEL.WEIGHTS = os.path.join("foam\_mask\_rcnn/" + cfg.OUTPUT\_DIR, "model\_final.pth")

    cfg.MODEL.ROI\_HEADS.SCORE\_THRESH\_TEST = 0.2

    predictor = DefaultPredictor(cfg)

    outputs = predictor(image)

    output\_instance = outputs["instances"].to("cpu")

    v = Visualizer(image[:,:,::-1], metadata={}, scale =1, instance\_mode=ColorMode.SEGMENTATION)

    v = v.draw\_instance\_predictions(output\_instance)

    detected\_classes = output\_instance.pred\_classes.numpy().tolist()

    # print(detected\_classes)

    bbox = None

    if detected\_classes != []:

        try:

            foam\_idx = detected\_classes.index(1)

        except ValueError: # if there is no foam

            return None, cv2.cvtColor(v.get\_image(), cv2.COLOR\_RGB2BGR)

        else:

            detected\_bboxs = output\_instance.pred\_boxes.tensor.numpy().tolist()

            bbox = detected\_bboxs[foam\_idx]

    # mask = output\_instance.pred\_masks.numpy()

    # print( mask.shape)

    return bbox, cv2.cvtColor(v.get\_image(), cv2.COLOR\_RGB2BGR)

## getDigestateInfo.py

Author(s): Ling Chen

from utils.getEdges import getEdges

import cv2

def getDigestateInfo(image, box, foam\_bbox, no\_foam):

    """

    Function to get various info on the digestate sample.

    :param image: image of sample

    :param box: bounding box coordinates of the flask

    :param foam\_bbox: bounding box of foam given in coordinates of top left and bottom right corners

    :param no\_foam: Boolean of whether there is foam or not

    :returns: a dictionary containing various information of the sample

    """

    canny = getEdges(image, 5)

    # cv2.imshow("edges for digestate info", canny)

    x,y,w,h = box

    # print(box)

    # plastic flask

    # TRUE\_FLASK\_HEIGHT = 0.1421 #metres

    # glass jar

    TRUE\_FLASK\_HEIGHT = 0.135 #metres

    # print( box)

    # cv2.imshow("c", canny)

    # cv2.waitKey(0)

    if no\_foam:

        foam\_height = None

        foam\_colour = None

        ratio\_foam\_to\_flask = 0

        # get liquid height

        for i in range(h):

            if canny[y+h-(i+1)][x+(w//2)] == 255 and i>50:

                liquid\_height = i

                break

        liquid\_colour = image[y+h-(liquid\_height//2)][x+(w//2)]

    else:

        foam\_height = round(abs(foam\_bbox[3] - foam\_bbox[1]))

        foam\_colour = image[round(abs(foam\_bbox[3]-foam\_bbox[1])//2)+round(min(foam\_bbox[1], foam\_bbox[3]))][x+(w//2)]

        ratio\_foam\_to\_flask = foam\_height/h

        # get liquid height

        foam\_edge = round(max(foam\_bbox[3], foam\_bbox[1]))

        # print(foam\_edge)

        # print(y)

        # print(h)

        # print(foam\_edge)

        liquid\_height = y+h-foam\_edge

        liquid\_colour = image[foam\_edge+(liquid\_height//2)][x+(w//2)]

    ratio\_liquid\_to\_flask = liquid\_height/h

    digestate\_info = {

        "digestate height": liquid\_height,

        "digestate colour": liquid\_colour,

        "foam height": foam\_height,

        "foam colour": foam\_colour,

        "real foam height": ratio\_foam\_to\_flask\*TRUE\_FLASK\_HEIGHT,

        "real digestate height": ratio\_liquid\_to\_flask\*TRUE\_FLASK\_HEIGHT

    }

    return digestate\_info

## colourBlockDetection.py

Author(s): Ling Chen

# Python code for Multiple Color Detection

import numpy as np

import cv2

def colourBlockDetection(imageFrame, viz, bgr):

    # Convert the imageFrame in

    # BGR(RGB color space) to

    # HSV(hue-saturation-value)

    # color space

    hsvFrame = cv2.cvtColor(imageFrame, cv2.COLOR\_BGR2HSV)

    bgr = np.uint8([[bgr]])

    hsv = cv2.cvtColor(bgr, cv2.COLOR\_BGR2HSV)

    hsv = np.squeeze(np.squeeze(hsv))

    # Set range for red color and

    # define mask

    h\_thresh = 50

    s\_thresh = 70

    v\_thresh = 70

    minHSV = np.array([hsv[0] - h\_thresh, hsv[1] - s\_thresh, hsv[2] - v\_thresh])

    if minHSV[0] <= 0:

        minHSV[0] = 1

    if minHSV[1] <= 0:

        minHSV[1] = 1

    if minHSV[2] <= 0:

        minHSV[2] = 1

    maxHSV = np.array([hsv[0] + h\_thresh, hsv[1] + s\_thresh, hsv[2] + v\_thresh])

    mask = cv2.inRange(hsvFrame, minHSV, maxHSV)

    # Morphological Transform, Dilation

    # for each color and bitwise\_and operator

    # between imageFrame and mask determines

    # to detect only that particular color

    kernal = np.ones((5, 5), "uint8")

    # For red color

    mask = cv2.erode(mask, kernal)

    mask = cv2.dilate(mask, kernal, iterations=2)

    res = cv2.bitwise\_and(imageFrame, imageFrame, mask = mask)

    # cv2.imshow("colour\_mask", mask)

    # print(hsv)

    # print(minHSV)

    # print(maxHSV)

    # Creating contour of colour blob

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

    max\_contour = sorted(contours, key = cv2.contourArea, reverse=True)[0]

    if cv2.contourArea(max\_contour) > 300:

        x, y, w, h = cv2.boundingRect(max\_contour)

        viz = cv2.rectangle(viz, (x, y), (x + w, y + h), (255, 0, 0), 2)

        # cv2.putText(viz, "Red Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 1.0, (0, 0, 255))

    return viz

## client\_pi.py

Author(s): Ling Chen

import os, socket, time, struct

HOST = "192.168.246.239" # Ling home wifi

HOST = "192.168.234.172" # Ling Hotspot

HOST = "172.20.10.8"  # Kabir Hotspot

PORT = 52212

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

#trying to connect to socket

# try:

sock.connect((HOST,PORT))

# Raspberry Pi sends over img it captures from pi-cam:

file\_name = "foam\_img.jpg"

file\_len = os.stat(file\_name).st\_size

msg\_header = struct.pack('<I', file\_len)

sock.sendall(msg\_header)

fd = open(file\_name, 'rb')

data = fd.read(file\_len)

while data:

    sock.sendall(data)

    data = fd.read(file\_len)

fd.close()

# print("Image Sent To Host")

viz\_list\_str = ["viz0000", "viz0001", "viz0010", "viz0011", "viz0100", "viz0101", "viz0110", "viz0111", "viz1000", "viz1001", "viz1010", "viz1011", "viz1100", "viz1101", "viz1110", "viz1111"]

# Client recieves processed img from server

for i in range(16):

    file\_name = 'transferred\_files/' + viz\_list\_str[i] + '.jpg'

    msg\_header = sock.recv(4)

    while len(msg\_header) != 4:

        msg\_header += sock.recv(4- len(msg\_header))

    file\_len = struct.unpack('<I', msg\_header)[0]

    nFile = open(file\_name, 'wb')

    data = sock.recv(file\_len)

    while len(data) != file\_len:

        data += sock.recv(file\_len - len(data))

    nFile.write(data)

    nFile.close()

    # print("Image Recieved From Host")

file\_name = "transferred\_files/digestate\_colour.jpg"

msg\_header = sock.recv(4)

while len(msg\_header) != 4:

    msg\_header += sock.recv(4- len(msg\_header))

file\_len = struct.unpack('<I', msg\_header)[0]

nFile = open(file\_name, 'wb')

data = sock.recv(file\_len)

while len(data) != file\_len:

    data += sock.recv(file\_len - len(data))

nFile.write(data)

nFile.close()

file\_name = "transferred\_files/foam\_colour.jpg"

msg\_header = sock.recv(4)

while len(msg\_header) != 4:

    msg\_header += sock.recv(4- len(msg\_header))

file\_len = struct.unpack('<I', msg\_header)[0]

nFile = open(file\_name, 'wb')

data = sock.recv(file\_len)

while len(data) != file\_len:

    data += sock.recv(file\_len - len(data))

nFile.write(data)

nFile.close()

sock.close()

## server\_host.py

Author(s): Ling Chen

import os, socket, time

from foam\_detection.process\_img import process\_img

import cv2

import struct

# Creating socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

HOST = "0.0.0.0"

PORT = 52212

machine = socket.gethostbyname(socket.gethostname())

sock.bind((HOST, PORT))

sock.listen(3)

print("HOST: ", sock.getsockname())

print("Server is Listening...")

# Accepting the connection from the client

client, addr = sock.accept()

# Server recieves img from raspberry pi

# Client recieves processed img from server

file\_name = 'transferred\_files/foam\_img\_from\_pi.jpg'

msg\_header = client.recv(4)

while len(msg\_header) != 4:

    msg\_header += client.recv(4- len(msg\_header))

file\_len = struct.unpack('<I', msg\_header)[0]

nFile = open(file\_name, 'wb')

data = client.recv(file\_len)

while len(data) != file\_len:

    data += client.recv(file\_len - len(data))

nFile.write(data)

nFile.close()

print("Image Recieved From Pi")

# Server processes img

img = cv2.imread("transferred\_files/foam\_img\_from\_pi.jpg")

viz\_list, viz\_list\_str, digestate\_colour, foam\_colour, digestate\_info = process\_img(img)

for i in range(len(viz\_list)):

    cv2.imwrite("transferred\_files/" +viz\_list\_str[i] + ".jpg", viz\_list[i])

cv2.imwrite("transferred\_files/digestate\_colour.jpg", digestate\_colour)

cv2.imwrite("transferred\_files/foam\_colour.jpg", foam\_colour)

# Server sends processed img back to pi

for i in range(len(viz\_list)):

    file\_name = "transferred\_files/" + viz\_list\_str[i] + ".jpg"

    file\_len = os.stat(file\_name).st\_size

    msg\_header = struct.pack('<I', file\_len)

    client.sendall(msg\_header)

    fd = open(file\_name, 'rb')

    data = fd.read(file\_len)

    while data:

        client.sendall(data)

        data = fd.read(file\_len)

    fd.close()

    print("Analyzed Image Sent To Client")

file\_name = "transferred\_files/digestate\_colour.jpg"

file\_len = os.stat(file\_name).st\_size

msg\_header = struct.pack('<I', file\_len)

client.sendall(msg\_header)

fd = open(file\_name, 'rb')

data = fd.read(file\_len)

while data:

    client.sendall(data)

    data = fd.read(file\_len)

fd.close()

print("Analyzed Image Sent To Client")

file\_name = "transferred\_files/foam\_colour.jpg"

file\_len = os.stat(file\_name).st\_size

msg\_header = struct.pack('<I', file\_len)

client.sendall(msg\_header)

fd = open(file\_name, 'rb')

data = fd.read(file\_len)

while data:

    client.sendall(data)

    data = fd.read(file\_len)

fd.close()

print("Analyzed Image Sent To Client")

client.close()

## maskRCNN\_training.py

Author(s): Ling Chen

import torch

import torchvision

import cv2

import os

import numpy as np

import json

import random

import matplotlib.pyplot as plt

%matplotlib inline

# pip install 'git+https://github.com/facebookresearch/detectron2.git'

from detectron2.structures import BoxMode

from detectron2.data import DatasetCatalog, MetadataCatalog

from detectron2.utils.logger import setup\_logger

from detectron2.data.datasets import register\_coco\_instances

from detectron2 import model\_zoo

from detectron2.engine import DefaultTrainer, DefaultPredictor

from detectron2.config import get\_cfg

from detectron2.utils.visualizer import ColorMode, Visualizer

from detectron2.utils.visualizer import Visualizer

from detectron2.utils.visualizer import ColorMode

import cv2

import pickle

#### Utils ####

def plot\_samples(dataset\_name, n=1):

    dataset\_custom = DatasetCatalog.get(dataset\_name)

    dataset\_custom\_metadata = MetadataCatalog.get(dataset\_name)

    for s in random.sample(dataset\_custom, n):

        img = cv2.imread(s["file\_name"])

        v = Visualizer(img[:,:,::-1], metadata=dataset\_custom\_metadata, scale = 1)

        v = v.draw\_dataset\_dict(s)

        plt.figure(figsize=(5,5))

        plt.imshow(v.get\_image())

        plt.show()

def get\_train\_cfg(config\_file\_path, checkpoint\_url, train\_dataset\_name, test\_dataset\_name, num\_classes, device, output\_dir):

    cfg = get\_cfg()

    cfg.merge\_from\_file(model\_zoo.get\_config\_file(config\_file\_path))

    cfg.MODEL.WEIGHTS = model\_zoo.get\_checkpoint\_url(checkpoint\_url)

    cfg.DATASETS.TRAIN = (train\_dataset\_name,)

    cfg.DATASETS.TEST = (test\_dataset\_name,)

    cfg.DATALOADER.NUM\_WORKERS = 2

    cfg.SOLVER.IMS\_PER\_BATCH = 5

    cfg.SOLVER.BASE\_LR = 0.00025

    cfg.SOLVER.MAX\_ITER = 3000

    cfg.SOLVER.STEPS = []

    cfg.MODEL.ROI\_HEADS.NUM\_CLASSES = num\_classes

    cfg.MODEL.DEVICE = device

    cfg.OUTPUT\_DIR = output\_dir

    return cfg

def on\_image(image\_path, predictor):

    im = cv2.imread(image\_path)

    outputs = predictor(im)

    print(outputs)

    v = Visualizer(im[:,:,::-1], metadata={}, scale =1, instance\_mode=ColorMode.SEGMENTATION)

    v = v.draw\_instance\_predictions(outputs["instances"].to("cpu"))

    plt.figure(figsize=(5,5))

    plt.imshow(v.get\_image())

    plt.show()

## If doing instance segmentation:

config\_file\_path = "COCO-InstanceSegmentation/mask\_rcnn\_R\_50\_FPN\_3x.yaml"

checkpoint\_url = "COCO-InstanceSegmentation/mask\_rcnn\_R\_50\_FPN\_3x.yaml"

output\_dir = "./output"

num\_classes = 2

device = "cpu"

train\_dataset\_name = "LP\_train"

train\_images\_path = "train"

train\_json\_annot\_path = "train/\_annotations.coco.json"

test\_dataset\_name = "LP\_valid"

test\_images\_path = "valid"

test\_json\_annot\_path = "valid/\_annotations.coco.json"

cfg\_save\_path = "IS\_cfg\_cpu.pickle"

register\_coco\_instances(name = train\_dataset\_name, metadata={}, json\_file=train\_json\_annot\_path, image\_root=train\_images\_path)

register\_coco\_instances(name = test\_dataset\_name, metadata={}, json\_file=test\_json\_annot\_path, image\_root=test\_images\_path)

# plot\_samples(train\_dataset\_name)

cfg = get\_train\_cfg(config\_file\_path, checkpoint\_url, train\_dataset\_name, test\_dataset\_name, num\_classes, device, output\_dir)\

with open(cfg\_save\_path, 'wb') as f:

    pickle.dump(cfg, f, protocol=pickle.HIGHEST\_PROTOCOL)

os.makedirs(cfg.OUTPUT\_DIR, exist\_ok=True)

trainer = DefaultTrainer(cfg)

trainer.resume\_or\_load(resume = False)

trainer.train()

cfg\_save\_path = "IS\_cfg\_cpu.pickle"

with open(cfg\_save\_path, 'rb') as f:

    cfg = pickle.load(f)

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT\_DIR, "model\_final.pth")

cfg.MODEL.ROI\_HEADS.SCORE\_THRESH\_TEST = 0.3

print(cfg.MODEL.DEVICE)

predictor = DefaultPredictor(cfg)

def test\_output(dataset\_name, n=1):

    dataset\_custom = DatasetCatalog.get(dataset\_name)

    dataset\_custom\_metadata = MetadataCatalog.get(dataset\_name)

    for s in random.sample(dataset\_custom, n):

        on\_image(s["file\_name"], predictor)

# test\_output(test\_dataset\_name, 5)

image\_path = "fg3.jpg"

on\_image(image\_path, predictor)

# Remote Control Web-app

## index.php

Author(s): Kabir Chugh, Ling Chen

<?php

    if (isset($\_POST['next'])){

        header("Location: main.php");

        exit();

    }

?>

<html>

<head><link rel = "stylesheet" href = "main.css"></head>

<div class = "top ">

    <font><center><b> <p class = "header\_text">Welcome to Team 232 Demonstration</p><b></center></font>

</div>

<div class = "below\_top">

</div>

<br><br><br>

<center>

<img src = "device.jpg" width="800" height="1000"  alt="Team Logo">

<br><br><br>

</div>

<div class = "below\_top">

    </div>

<div class = "below\_top">

<form method="post">

    <button name = "next">

    <p class ="button\_text">Start Sampling</p>

    </button>

</form>

<br>

</font>

<script>

</script>

</html>

## main.php

Author(s): Kabir Chugh, Ling Chen

<?php

    $weight = "---";

    $warning\_message = "";

    if (isset($\_POST['b7']))

    {

        $warning\_message = "Calculating Weight...";

        // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_x = intval($tilt\_x);

        $tilt\_y = intval($tilt\_y);

        $tilt\_inbound\_x = ($tilt\_x <15 and $tilt\_x >-15);

        $tilt\_inbound\_y = ($tilt\_y <15 and $tilt\_y >-15);

        $tilt\_inbound = $tilt\_inbound\_x and $tilt\_inbound\_y;

        if(! $tilt\_inbound)

        {

            $weight = "---";

            $warning\_message = "Plaform is tilted!";

        } else

        {

            // Get weight using scripts

            shell\_exec('python3 MAIN2.py');

            $file = fopen("weights.txt","r");

            $weight = fread($file, filesize("weights.txt"));

            fclose($file);

            $valid\_weight = True;

            foreach (str\_split($weight) as $char) {

                if (($char != ".")and ($char != "0") and ($char != "1") and ($char != "2") and ($char != "3") and ($char != "4") and ($char != "5") and ($char != "6") and ($char != "7") and ($char != "8") and ($char != "9")){

                    $valid\_weight = False;

                    break;

                }

            }

            $weight\_inbound = $weight>140 and $weight<160;

            if ($valid\_weight){

                if ($weight\_inbound)

                {

                    $warning\_message = "Weight is within bounds";

                } else

                {

                    $warning\_message = "The weight of the given sample is out of bounds of 140g and 160g.";

                }

            }

            else {

                $warning\_message = $weight;

                $weight = "---";

            }

        }

    }

    if (isset($\_POST['next'])){

                // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_x = intval($tilt\_x);

        $tilt\_y = intval($tilt\_y);

        $tilt\_inbound\_x = ($tilt\_x <15 and $tilt\_x >-15);

        $tilt\_inbound\_y = ($tilt\_y <15 and $tilt\_y >-15);

        $tilt\_inbound = $tilt\_inbound\_x and $tilt\_inbound\_y;

            if ($tilt\_inbound){

                // Get weight using scripts

                shell\_exec('python3 MAIN2.py');

                $file = fopen("weights.txt","r");

                $weight = fread($file, filesize("weights.txt"));

                fclose($file);

                $valid\_weight = True;

                foreach (str\_split($weight) as $char) {

                    if (($char != ".")and ($char != "0") and ($char != "1") and ($char != "2") and ($char != "3") and ($char != "4") and ($char != "5") and ($char != "6") and ($char != "7") and ($char != "8") and ($char != "9")){

                        $valid\_weight = False;

                        break;

                    }

                }

                $weight\_inbound = $weight>140 and $weight<160;

                if ($weight\_inbound){

                    header("Location: servo.php");

                    exit();

                } else {

                    $warning\_message = "Make sure weight is within 140-160 grams before proceeding!";

                }

            } else {

                $warning\_message = "Make sure platfrom is flat before proceeding!";

            }

    }

?>

<html>

<head>

<link rel = "stylesheet" href = "main.css">

</head>

<body>

<div class = "top ">

    <font><center><b> <p class = "header\_text">Measure Weight</p><b></center></font>

</div>

<div class = "below\_top">

</div>

<div class = "below\_top">

</div>

<div class = "below\_top">

</div>

<div class = "below\_top">

</div>

    <center>

    <form method = "post">

    <button name = "b7" id="function\_btn">

        <p class ="button\_text">Display weight </p>

    </button>

    <br><br>

    <p>

    The weight of the given sample is:

    </p>

    <div class = "separator"></div>

    <p class = "sec">

     <?=$weight ?> grams

    </p>

    <div class = "separator"></div>

    <p class = "warning">

    <?=$warning\_message ?>

    </p>

    <br><br><br>

    <br>

    <button name="next">

        <p class ="button\_text">Next</p>

    </button>

    </form>

    </center>

<script type = "text/javascript" >

</script>

</body>

</html>

## servo.php

Author(s): Kabir Chugh, Ling Chen

<?php

    $img = "";

    $warning\_message = "";

    if (isset($\_POST['b1'])){

        // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                $warning\_message = "";

                shell\_exec('python3 Servo\_0.py');

            } else {

                $warning\_message = "Platform is too tilted!";

            }

        }

    if (isset($\_POST['b2'])){

                    // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                $warning\_message = "";

                shell\_exec('python3 Servo\_45.py');

            } else {

                $warning\_message = "Platform is too tilted!";

            }

        }

    if (isset($\_POST['b3'])){

                    // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                $warning\_message = "";

                shell\_exec('python3 Servo\_90.py');

            } else {

                $warning\_message = "Platform is too tilted!";

            }

        }

    if (isset($\_POST['b4'])){

                    // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                shell\_exec('python3 Servo\_135.py');

            } else {

                $warning\_message = "Platform is too tilted!";

            }

        }

    if (isset($\_POST['b5'])){

                    // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                $warning\_message = "";

                shell\_exec('python3 Servo\_180.py');

            } else {

                $warning\_message = "Platform is too tilted!";

            }

        }

    if(True){

        // shell\_exec('python3 camera\_image.py');

        // $file = fopen("img\_num.txt","r");

        // $img\_num= fread($file, filesize("img\_num.txt"));

        //fclose($file);

        //$file\_name = "images/foam\_img".$img\_num.".png";

        sleep(1);

        $img = '<center> <img src = foam\_img.jpg alt = "Sample Image"></center>';

    }

    if (isset($\_POST['pic'])){

                // Get tilt using script

        shell\_exec('python3 MPU.py');

        $file = fopen("angle\_x.txt","r");

        $tilt\_x = fread($file, filesize("angle\_x.txt"));

        fclose($file);

        $file = fopen("angle\_y.txt","r");

        $tilt\_y = fread($file, filesize("angle\_y.txt"));

        fclose($file);

        $tilt\_inbound = ($tilt\_x <15 and $tilt\_x >-15) and ($tilt\_y <15 and $tilt\_y >-15);

            if ($tilt\_inbound){

                $warning\_message = "Analyzing...";

                shell\_exec('python3 client\_pi.py');

                header("Location: foam.php");

                exit();

            } else {

                $warning\_message = "Platform is too tilted!";

            }

    }

?>

<html>

<head>

<link rel = "stylesheet" href = "main.css">

</head>

<body>

<div class = "top ">

    <font><center><b> <p class = "header\_text">Capture Image</p><b></center></font>

</div>

<div class = "below\_top">

</div>

    <center>

    <?= $img ?>

    <br><br>

        <div class = "separator"></div>

    <form method = "post">

<p>Select the angle for servo rotation:</p>

    <div class = "button\_row">

    <button name = "b1" id="function\_btn1">

        <p class ="button\_text"> 0 &nbsp &nbsp Degrees </p>

    </button>

    &nbsp &nbsp

    <button name = "b2" id="function\_btn2">

    <p class ="button\_text">    45 &nbsp Degrees </p>

    </button>

    </div>

    <div class = "button\_row">

    <button name = "b3" id="function\_btn1" >

        <p class ="button\_text">90 &nbsp Degrees</p>

    </button>

    &nbsp &nbsp

    <button name = "b4" id="function\_btn2">

        <p class ="button\_text">135 &nbsp Degree</p>

    </button>

    </div>

    <div class = "button\_row">

    <button name = "b5" id="function\_btn3">

        <p class ="button\_text">180 Degrees</p>

    </button>

    </div>

    <br><br>

    <div class = "separator"></div>

    <br><br>

    <button name = "pic" onClick = redirectToMain()>

        <p class ="button\_text">Analyse Image</p>

    </button>

    <br><br>

    <?=$warning\_message ?>

    </form>

    </center>

<script type = "text/javascript" >

    function redirectToMain()

    {

        window.location.replace("foam.php")

    }

</script>

</body>

</html>

## foam.php

Author(s): Kabir Chugh, Ling Chen

<?php

    $digestate =  '<img src = transferred\_files/digestate\_colour.jpg alt = "digestate colour" style="width:300px; height:100px; margin-left:125px">';

    $foam =  '<img src = transferred\_files/foam\_colour.jpg alt = "foam colour" style="width:300px; height:100px; margin-left:125px">';

    if (isset($\_POST['pic'])){

        header("Location: servo.php");

        exit();

    }

    if (isset($\_POST['foam'])){

        if (isset($\_POST['foam\_height'])){

            if (isset($\_POST['digestate'])){

                if (isset($\_POST['digestate\_height'])){

                    //1111

                    $img = '<img src = transferred\_files/viz1111.jpg alt = "Sample Image">';

                } else {

                    //1110

                    $img =  '<img src = transferred\_files/viz1110.jpg alt = "Sample Image">';

                }

            } else {

                if (isset($\_POST['digestate\_height'])){

                    //1101

                    $img =  '<img src = transferred\_files/viz1101.jpg alt = "Sample Image">';

                } else {

                    //1100

                    $img =  '<img src = transferred\_files/viz1100.jpg alt = "Sample Image">';

                }

            }

        } else {

            if (isset($\_POST['digestate'])){

                if (isset($\_POST['digestate\_height'])){

                    //1011

                    $img =  '<img src = transferred\_files/viz1011.jpg alt = "Sample Image">';

                } else {

                    //1010

                    $img =  '<img src = transferred\_files/viz1010.jpg alt = "Sample Image">';

                }

            } else {

                if (isset($\_POST['digestate\_height'])){

                    //1001

                    $img =  '<img src = transferred\_files/viz1001.jpg alt = "Sample Image">';

                } else {

                    //1000

                    $img =  '<img src = transferred\_files/viz1000.jpg alt = "Sample Image">';

                }

            }

        }

    } else {

        if (isset($\_POST['foam\_height'])){

            if (isset($\_POST['digestate'])){

                if (isset($\_POST['digestate\_height'])){

                    //0111

                    $img =  '<img src = transferred\_files/viz0111.jpg alt = "Sample Image">';

                } else {

                    //0110

                    $img =  '<img src = transferred\_files/viz0110.jpg alt = "Sample Image">';

                }

            } else {

                if (isset($\_POST['digestate\_height'])){

                    //0101

                    $img =  '<img src = transferred\_files/viz0101.jpg alt = "Sample Image">';

                } else {

                    //0100

                    $img =  '<img src = transferred\_files/viz0100.jpg alt = "Sample Image">';

                }

            }

        } else {

            if (isset($\_POST['digestate'])){

                if (isset($\_POST['digestate\_height'])){

                    //0011

                    $img =  '<img src = transferred\_files/viz0011.jpg alt = "Sample Image">';

                } else {

                    //0010

                    $img =  '<img src = transferred\_files/viz0010.jpg alt = "Sample Image">';

                }

            } else {

                if (isset($\_POST['digestate\_height'])){

                    //0001

                    $img =  '<img src = transferred\_files/viz0001.jpg alt = "Sample Image">';

                } else {

                    //0000

                    $img =  '<img src = transferred\_files/viz0000.jpg alt = "Sample Image">';

                }

            }

        }

    }

?>

<html>

<head>

<link rel = "stylesheet" href = "main.css">

</head>

<body>

<div class = "top ">

    <font><center><b> <p class = "header\_text">Analysis Results</p><b></center></font>

</div>

<div class = "below\_top">

</div>

    <center><?=$img?></center>

<div class = "below\_top">

</div>

    <div style="width:100%; overflow: hidden;">

        <div style = "width:500px; float: left;">

            <p class = "colour"> Foam Colour: </p>

            <?=$foam?>

            <br>

            <p class = "colour"> Digestate Colour: </p>

            <?=$digestate?>

        </div>

        <div style = "margin-left: 550px; margin-top: 0;">

            <p> Display Options: </p>

            <form method = "post">

            <input type ="checkbox" name = "foam" >

            <label for ="foam" class = "cform">Foam Area</label><br><br>

            <input type ="checkbox" name = "foam\_height">

            <label for ="foam\_height" class = "cform">Foam Height</label><br><br>

            <input type ="checkbox" name = "digestate">

            <label for ="digestate" class = "cform">Digestate Area</label><br><br>

            <input type ="checkbox" name = "digestate\_height">

            <label for ="digestate\_height" class = "cform">Digestate Height</label><br><br>

            <br>

            <button name = "submit" onClick = submit()>

                <p class ="button\_text">Submit</p>

            </button> <br>

        </div>

    </div>

    <center>

        <br><br>

        <div class = "separator"></div>

        <br><br>

    <button name = "pic">

        <p class ="button\_text">Retake Image</p>

    </button>

    </form>

    </center>

</body>

</html>

<script>

/\*

document.addEventListener('DOMContentLoaded',()=>{

    let chk=document.querySelector('input[type="checkbox"][name="foam"]');

        chk.checked=localStorage.getItem( chk.name )==null || localStorage.getItem( chk.name )=='false' ? false : true;

        chk.addEventListener('click',e=>{

            localStorage.setItem( chk.name, chk.checked )

            //location.reload();

        });

});

function set\_state(){

    \*/

}

</script>

## main.css

Author(s): Kabir Chugh, Ling Chen

button{

font-size: 30px;

transition-duration: 0.4s;

background-color: #242526;

border: 2px solid #242526;

border-radius: 6px;

padding: 5px 100px 0 100px ;

}

button:hover{

background-color: #3a3b3c;

}

p{

font-size: 40px;

color: #E4E6EB;

font-weight: normal;

font-family: 'Arial';

}

.button\_text{

font-size: 30px;

color: #E4E6EB;

}

body{

background-color: #18191A;

}

.header\_text{

font-size: 50px;

font-weight: normal;

font-family: 'Arial';

color: #E4E6EB;

padding: 0 20px 0 20px;

}

.header\_text2{

font-size: medium;

font-weight: bold;

font-family: 'Courier New', Courier, monospace;

color: rgb(0,255,255);

}

.header\_text3{

font-size: medium;

font-weight: bold;

font-family: 'Courier New', Courier, monospace;

color: rgb(0,255,255);

}

.header\_text4{

font-size: medium;

font-weight: bold;

font-family: 'Courier New', Courier, monospace;

color: rgb(0,255,255);

}

.container{

display: flex;

justify-content: space-around;

}

.top{

    padding: 50px 0;

    background-color: #242526;

}

.below\_top{

    padding: 25px 0;

}

.sec{

    font-size: 40px;

    color: #BOB3B8;

}

.warning{

    font-size: 35px;

    color: #7f0000;

    padding: 0 20px 0 20px;

}

.separator{

    background-color: #242526;

    padding: 0 100px 0 100px;

    height: 10px

}

.button\_row{

    padding: 10px 0 10px 0;

}

.colour{

    font-size: 35px;

    margin-left: 125px;

}

.cform{

    font-size: 35px;

    font-weight: normal;

    font-family: 'Arial';

    color: #E4E6EB;

}

input.largerCheckbox {

    width: 30px;

    height:30px;

}

## Servo[0-180].py

Author(s): Jason Ltg

from gpiozero import Servo

import math

from time import sleep

from gpiozero.pins.pigpio import PiGPIOFactory

factory = PiGPIOFactory()

servo = Servo(17, min\_pulse\_width = 0.5/1000, max\_pulse\_width = 2.33/1000, pin\_factory=factory)

try:

    angle = 180 #CHANGE THIS VALUE TO WHATEVER ANGLE NEEDED

    servo.value = math.cos(math.radians(angle)) \* -1

    sleep(1)

except KeyboardInterrupt:

    #Add extra here for safety upon "Ctrl + C" keyboard input exit

    print("Goodbye!")

Note: Other files needed for web-app are omitted as they are just slight variations of the code shown in sections 1 to 4.