Portable, low-cost, Raspberry Pi-based optical sensor (PiSENS): Continuous monitoring of atmospheric nitrogen dioxide

Ernesto Saiz*[1], Ivana Banicevic[2], Sergio Espinoza Torres[3], Gino Picasso[3] Matthew O'Brien*[1], Aleksandar Radu*[4]

[1]Lennard-Jones Laboratories, Birchall Centre, Keele University, Keele, Staffordshire, ST5 5BG, UK

[2]Faculty of Technical Sciences, University of Montenegro, Montenegro

[3]Laboratory of Physical Chemistry Research, Faculty of Sciences, National University of Engineering, Av. Tupac Amaru 210, Lima 25, Peru

School of Chemistry, Joseph Banks Laboratories, University of Lincoln, Green Lane, Lincoln, LN6 7DL, UK

ABSTRACT: We have developed a sensing system that utilizes a low-cost computer (Raspberry Pi) and its imaging camera as an optical sensing core for the continuous detection of NO_2 in the air (PiSENS-A). The sensor is based on colour development as a consequence of the interaction of the gas with an absorbing solution. The PiSENS-A is thoroughly calibrated over the hourly mean which is used as one of the key metrics in evaluating air quality. The calibration was performed in the range of $0 < [NO_2] < 476$ µg/m^3 chosen to contain the threshold used to determine compliance to the UK's Air Quality Standard Regulations (2010) expressed as a maximum of 18 permitted exceedance of $[NO_2]_{\text{hourly mean}} = 200$ µg/m^3 per year. Lab-based measurements were evaluated against UV-Vis. The average precision expressed as a relative standard deviation was: RSD% = 2.8%, while the correlation of mock samples was excellent (Pearson's r = 1.000). Field-based measurements were evaluated against chemiluminescence-based instrument exhibiting a correlation coefficient of $R^2 = 0.993$. The PiSENS-A was also deployed as an independent air quality analyser at the Keele University campus.

Supporting Information

Contents

Calibration of PiSENS

Background

For the calibration curve to be valid, the inverse function (mapping the measured projection to a function of concentration) must be single-valued. This places limitations on the topology of the [R, G, B] curve as it varies with concentration. In particular, the projection of the curve onto the base vector must be monotonic with respect to concentration, although significant curvature away from the base vector can be allowed.

A hypothetical example curve (which has significantly exaggerated curvature away from the base vector) is shown in Figure SI1 left. The measured values of [R, G, B] for a series of samples of increasing concentration follow the trajectory shown in orange (actual data points are shown in blue). The base vector is represented by the blue line between the first and last data points. The projection of the vector resulting from each [R, G, B] position (this [R, G, B] position minus the position of the zero [R, G, B] position) with the base vector is represented by the red points on the base vector. The 'error' vectors (perpendicular to the projection vectors) connecting each sample [R, G, B] position with the projection on the base vector are shown in green. As can be seen, the value of the projection is monotonic with respect to movement along the [R, G, B] curve (we also assume that position on this curve is monotonic with respect to concentration). The graph of projection against concentration is shown in Figure SI1 right. Concentration is single-valued for all values of projection, satisfying the requirement for use as a calibration curve.

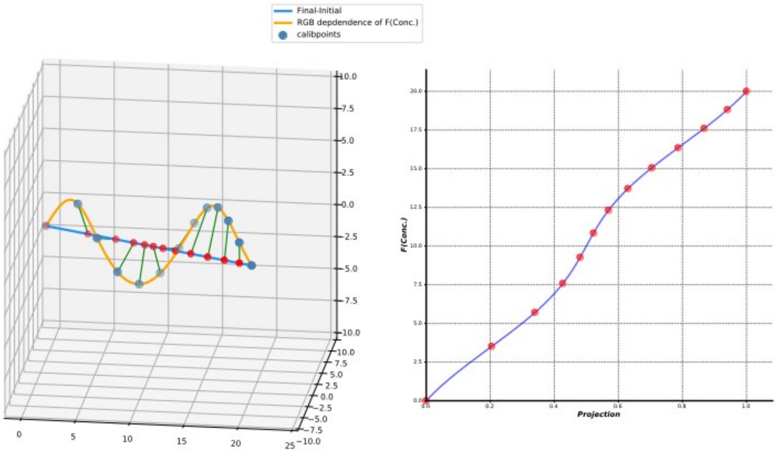

Figure SI1. Hypothetical calibration curve obtained using a series of calibration points (function F1). Left) Projection in RGB space: actual calibration data points (green dots), trajectory (orange line), base vector (blue line), projection of the vector from each [R, G, B] position ([R, G, B]$_{point}$ – [R, G, B]$_{zero}$)(red points on the base vector), the 'error' vectors (perpendicular to the projection vectors) connecting each sample [R, G, B] position with the projection on the base vector (green). Right) projections against concentrations.

Another example curve is shown in Figure SI2. In this case, the curve 'doubles back' on itself in several places. This means that the value of the projection on the base vector is not monotonic with respect to movement along the [R, G, B] curve (i.e. with concentration). This results in the mapping of projection to concentration being non-invertible. As can be seen in Figure SI2, concentration has multiple values in the two highlighted regions and this curve of [R, G, B] response would therefore not be allowed.

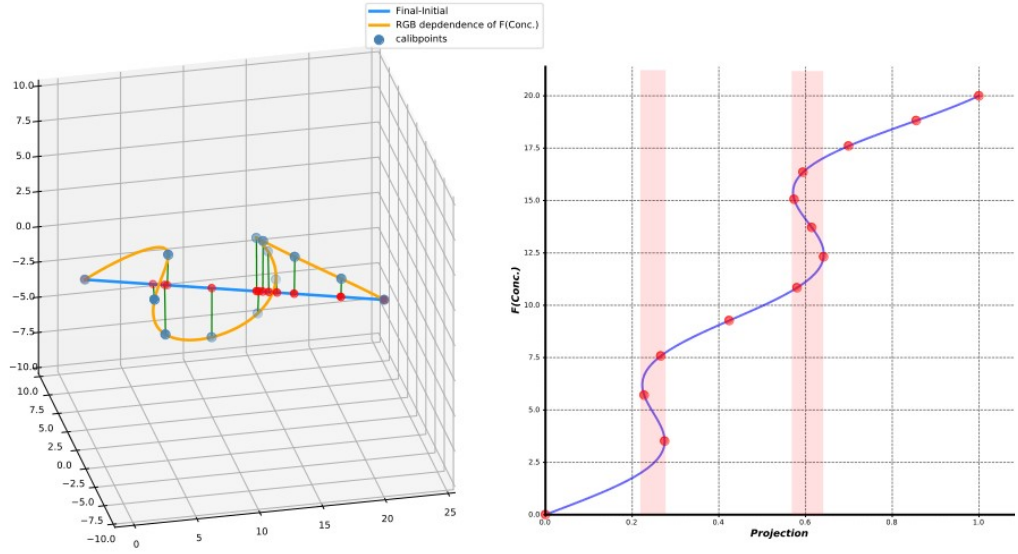

Figure SI2. Hypothetical calibration curve obtained using a series of calibration points (function F1). Left) Projection in RGB space: actual calibration data points (green dots), trajectory (orange line), base vector (blue line), projection of the vector from each [R, G, B] position ([R, G, B]$_{point}$ – [R, G, B]$_{zero}$)(red points on the base vector), the 'error' vectors (perpendicular to the projection vectors) connecting each sample [R, G, B] position with the projection on the base vector (green). Right) projections against concentrations.

Pre-calibration with reference papers

Before investigating the ability of our system to generate and use calibration curves to map colour intensity to actual NO_2 concentration, we first sought to establish that the system could reliably and repeatably provide colour intensities across an arbitrary range of 'reasonable' values.

We have prepared sets of 10 shades of blue, green, pink, and orange papers by printing as shown in Figure SI3 (the colour intensity was varied by changing the opacity of the coloured rectangle across the series so that the intensity increased monotonically). This was followed by the recording of the signal intensity of each shade as a scalar projection of its RGB vector (its [RGB] tuple minus the [RGB] tuple for the white paper) against the min/max RGB vector (the [RGB] tuple of the darkest shade minus the [RGB] tuple of the white paper). Figure SI4 illustrates signals obtained for the blue series. Note that there is no expectation of any quantifiable difference between the shades and that there is no expectation of any particular curve shape except that it is likely to be monotonic and the first and final values must be 0.0 and 1.0). For each shade of colour, we then took a further 10 measurements of colour intensity. The average value for each of the 10 measurements was then plotted against the initial measurement of colour intensity for that shade. In other words, we compare the averaged signal of 10 measurements against the initial recording. We expected a linear dependence (as this is simply a plot of the initial measurement against subsequent measurements of the same object) while allowing for minimal deviations due to differences in, for example, position, light, etc. Figure SI5 illustrates the evaluation graph.

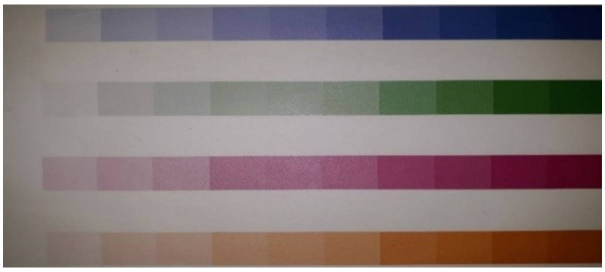

Figure SI3. Colour reference illustrating shades of blue, green, red, and orange printed on white paper

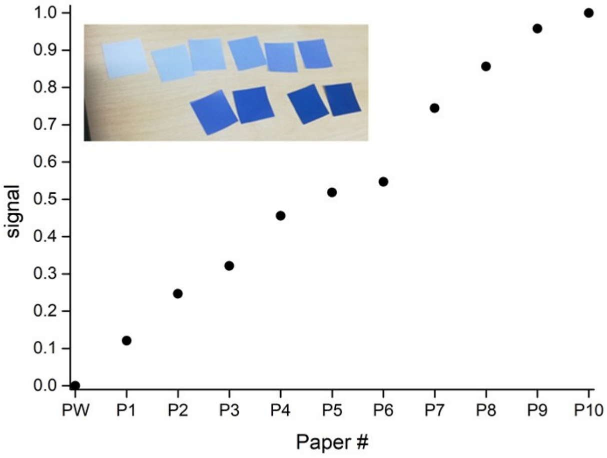

Figure SI4. Pre-*calibration using coloured paper. White paper (PW) was used as a reference and set as a minimum (0) and the darkest shade (P10) was set as the maximum. Intermediate points are obtained with shades P1 – P9. Error bars of triplicate measurements are smaller than the symbols. Inset: the image of used blue paper (white paper not shown).*

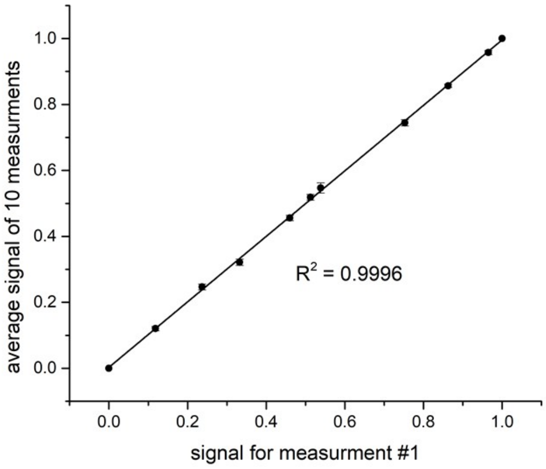

Figure SI5. Evaluation of repeatability measurements using blue paper by plotting the initial measurement for each shade against the average of 10 subsequent measurements for the same shade. Error bars show the standard deviation of n=10 measurements.

Calibration with [NO_2] standards

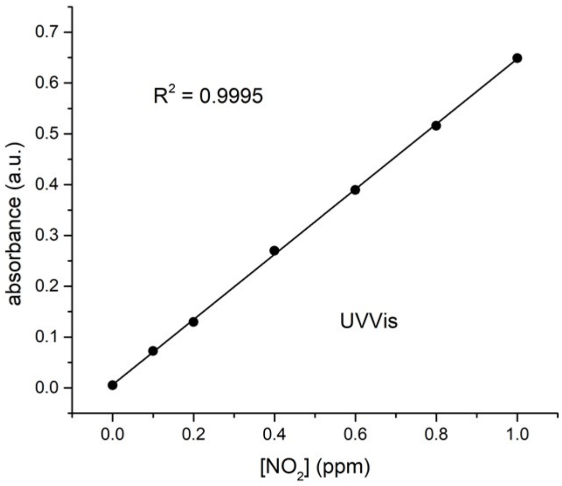

Figure SI6. Calibration using [NO_2] standard and UV-Vis. Error bars (n = 4) are smaller than the size of data symbols.

Continuous monitoring of [NO_x] in air

For comparison purposes, Figure SI7 shows [NO_x] obtained at Keele University campus as shown in the main paper (full lines) and at Stoke-on-Trent monitoring sites where DEFRA has air monitoring stations which are a part of Automatic Urban and Rural Monitoring Network (AURN).

The AURN (https://uk-air.defra.gov.uk/networks/network-info?view=aurn) is the UK's largest automatic monitoring network and is the main network used for compliance reporting against the Ambient Air Quality Directives. It includes automatic air quality monitoring stations measuring oxides of nitrogen (NO_x), sulphur dioxide (SO_2), ozone (O_3), carbon monoxide (CO) and particles (PM_{10}, $PM_{2.5}$). These sites provide high-resolution hourly information which is communicated rapidly to the public, using a wide range of electronic, media and web platforms.

The AURN uses chemiluminescence for the determination of NO/NO_2.

The data from AURN shown in Figure SI7 originate from two monitoring stations: one named Stoke-on-Trent Centre (UKA00337) which is located in a self-contained, air-conditioned housing situated in the northern part of the central business district of Hanley. The station lies approximately 5 metres from the approach road to a busy multi-storey car park which is located some 50 metres from the station. The second is named A50 Roadside (UKA00610), which is located next to a busy road, the A50 Potteries Way, in the Stoke-on-Trent borough of Meir. The station itself is located between the A50 and a parallel road that provides access to residences and businesses.

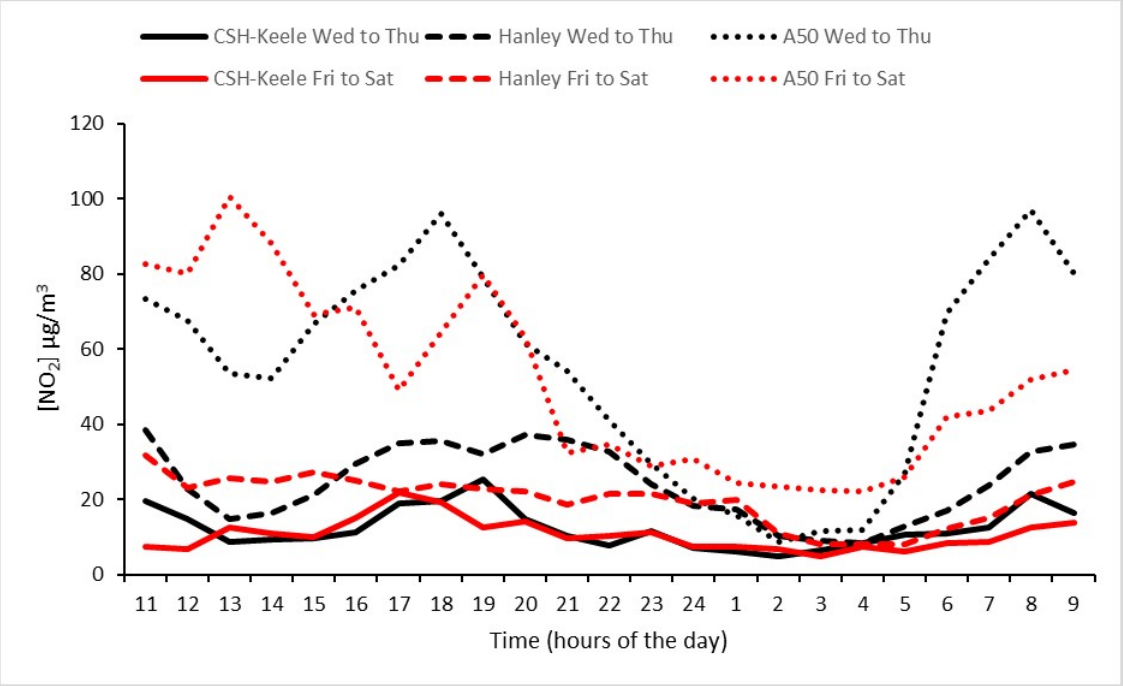

Figure SI7. Monitoring of [NO_x] in air. Full lines) [NO_x] obtained by PiSENS-A at Crime Scene House (CSH) at Keele University campus from 10 am 03rd – 9 am 04th February (Wednesday to Thursday) (black) and from 10 am 05th – 9 am 06th February 2021 (red). Dashed lines) [NO_x] obtained from AURN's Stoke-on-Trent Centre location, and dotted lines) [NO_x] was obtained from AURN's Stoke-on-Trent A50 location for the days when monitoring of [NO_x] was done by PiSENS-A at the Keele University campus.*

© Crown 2021 copyright Defra via uk-air.defra.gov.uk, licenced under the Open Government Licence (OGL).

Comparison of analytical characteristics of selected relevant methods for the determination of NO_2 in the atmosphere

Table SI1. Comparison of analytical characteristics between our method and several other methods used to determine NO_2 in the atmosphere.

Analytical method	**Sampler system**	**Sampling time range (h)**	**Linear range (from LOD) (µg/m^3)**	**Precision (RSD%) & accuracy (RE%)**	**Reference**
Optical sensor (Photodiode)	Micro air pump (0.05 L/min) – porous membrane	Up to 15 h	9.59 – 1880.5	4.1% 3.5%	[1]
Optical (digital image)	Silica/C-18 cartridges	1 h	15.0 - 100	3.2% 7.0%	[2]

Optical (digital image)	Silica/C-18 cartridges	1 h	5.0 – 54.0	1.6% 2.0%	[2]
Optical (spectrophotometry)	Diaphragm pump - Micro impinger	Up to 1 h	14	6.9% 12.0%	[3]
Optical (spectrophotometry)	Impregnated filters (passive)	Up to 288 h	1.1 - -	5.3% -	[4]
Optical (Colour sensor)	Wet pads	Up to 2 h	94 – 5.6x10^{5}	- -	[5]
Optical (Smartphone)	Pump – TEA impregnated pads.	1 h	4.8 - 191	9% -	[6]
Optical (Photodiode)	Pump – liquid reagent	1 h	1.88 – 376.1	2.2% 31.4%	[7]
Optical (portable spectrophotometry)	Pump - impinger	2 h	0.56 - 1880	3.6% -	[8]
Optical (PiSENS-A colour sensor)	Pump - impinger	0.3 h up to 72 h	0.034 - 1191	2.85% 6.64%	This work

(1) Tian, Y.; Zhang, X.; Shen, H.; Liu, A.; Zhao, Z.; Chen, M.-L.; Chen, X.-W. High Time-Resolution Optical Sensor for Monitoring Atmospheric Nitrogen Dioxide. *Anal. Chem.* **2017**, *89* (24), 13064–13068. https://doi.org/10.1021/acs.analchem.7b03578.

(2) Passaretti Filho, J.; da Silveira Petruci, J. F.; Cardoso, A. A. Development of a Simple Method for Determination of NO2 in Air Using Digital Scanner Images. *Talanta* **2015**, *140*, 73–80. https://doi.org/10.1016/j.talanta.2015.03.009.

(3) Filho, J. P.; Costa, M. A. M.; Cardoso, A. A. A Micro-Impinger Sampling Device for Determination of Atmospheric Nitrogen Dioxide. *Aerosol Air Qual. Res.* **2019**, *19* (11), 2597–2603. https://doi.org/10.4209/aaqr.2019.06.0318.

(4) Felix, E.; Gidhagen, L.; Alonso, M. F.; Nahirny, E. P.; Alves, B. L.; Segersson, D.; Amorim, J. H. Passive Sampling as a Feasible Tool for Mapping and Model Evaluation of the Spatial Distribution of Nitrogen Oxides in the City of Curitiba, Brazil. *Air Qual Atmos Health* **2019**, *12* (7), 837–846. https://doi.org/10.1007/s11869-019-00701-z.

(5) Fàbrega, C.; Fernández, L.; Monereo, O.; Pons-Balagué, A.; Xuriguera, E.; Casals, O.; Waag, A.; Prades, J. D. Highly Specific and Wide Range NO2 Sensor with Color Readout. *ACS Sens.* **2017**, *2* (11), 1612–1618. https://doi.org/10.1021/acssensors.7b00463.

(6) Cerrato-Alvarez, M.; Frutos-Puerto, S.; Arroyo, P.; Miró-Rodríguez, C.; Pinilla-Gil, E. A Portable, Low-Cost, Smartphone Assisted Methodology for on-Site Measurement of NO2 Levels in Ambient Air by Selective Chemical Reactivity and Digital Image Analysis. *Sensors and Actuators B: Chemical* **2021**, *338*, 129867. https://doi.org/10.1016/j.snb.2021.129867.

(7) da Silva Sousa, D.; Leal, V. G.; dos Reis, G. T.; da Silva, S. G.; Cardoso, A. A.; da Silveira Petruci, J. F. An Automated, Self-Powered, and Integrated Analytical Platform for On-Line and In Situ Air Quality Monitoring. *Chemosensors* **2022**, *10* (11), 454. https://doi.org/10.3390/chemosensors10110454.

(8) Subba, J. R.; Thammakhet, C.; Thavarungkul, P.; Kanatharana, P. Distributions of SO2 and NO2 in the Lower Atmosphere of an Industrial Area in Bhutan. *Journal of Environmental Science and Health, Part A* **2016**, *51* (14), 1278–1287. https://doi.org/10.1080/10934529.2016.1215196.

Python script

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy as np
import numpy.linalg as lin
import scipy as sp
from scipy import interpolate
from matplotlib import pyplot as plt
import os


camera = PiCamera()
#camera.awb_mode='tungsten'
camera.awb_mode='off'
camera.awb_gains=(1.4, 2.25)
camera.exposure_mode='auto'
#camera.analog_gain=(2.50)
#camera.digital_gain=(1.0)
camera.resolution = (640, 480)
camera.framerate = 10

rawCapture = PiRGBArray(camera, size=(640,480))
```

```
time.sleep(0.1)

drag_start = None
drag_end = None
box = None
calibflag = 0



calibvals = []
calibconcs = [0, 0.1, 0.2, 0.4, 0.6, 0.8, 1]
calibconcs = np.array(calibconcs)

calibprojs = []

sampleconcs = []

samplergbs = []

sampleprojs = []

def onedvectproj(v1, v2):


    scalar = (np.dot(v1,v2))/((lin.norm(v2))**2)
    #vector = scalar*v2


    return scalar

def calibrate():
    global calibvals
    global calibflag
    global calibprojs
    global calibconcs
    calibvals = np.array(calibvals)
    maxvect = calibvals[-1] - calibvals[0]
    for calibval in calibvals:
        scalar = onedvectproj((calibval-calibvals[0]), maxvect)
        calibprojs.append(scalar)

    sorted_index = np.argsort(calibprojs)
    calibprojs = [calibprojs[i] for i in sorted_index]
    calibconcs = [calibconcs[i] for i in sorted_index]

    f3 = interpolate.PchipInterpolator(calibprojs, calibconcs)

    print('calibrate')
    calibflag = 1

    return f3

def getave(pic, box):



    a = box[0]
    b = box[1]
    c = box[2]
    d = box[3]
    region = pic[c:d, a:b]
    mean = cv2.mean(region)
    mean = mean[:3]
    #region = pic[a:b, c:d]
    return  mean

def drawrect(pic, box, col):

    a = box[0]
    b = box[1]
    c = box[2]
    d = box[3]
    #region = pic[c:d, a:b]

    cv2.rectangle(pic,(a,c),(b,d),(col),3)
```

```
def mousebox(event, x, y, flags, param):
    global drag_start
    global drag_end
    global box
    if event == cv2.EVENT_LBUTTONDOWN:
        drag_end = False
        drag_start = (x,y)
        print (drag_start)
        print ('left down')
    if event == cv2.EVENT_LBUTTONUP:
        drag_end = (x,y)
        print (drag_end)
        print ('left up')

    if drag_start and drag_end:


        xmin = min (drag_start[0], drag_end[0])
        ymin = min (drag_start[1], drag_end[1])
        xmax = max (drag_start[0], drag_end[0])
        ymax = max (drag_start[1], drag_end[1])
        box = (xmin, xmax, ymin, ymax)
    if box:
        print (box)

for frame in camera.capture_continuous(rawCapture, format='bgr', use_video_port = True):
    image = frame.array

    if box:
        drawrect(image, box, (255,255,0))
        ave = getave(image, box)
        #print(ave)

    cv2.imshow('Frame', image)
    cv2.setMouseCallback('Frame', mousebox)
    key = cv2.waitKey(10) & 0xFF
```

```
rawCapture.truncate(0)

if key == ord('q'):
   cv2.destroyAllWindows()
   break

if key == ord('c'):
   print('c pressed')
   if box:
      if len(calibvals)<len(calibconcs):
         calibvals.append(ave)
         print(str(len(calibvals)) + ' calibvals obtained')

if key == ord('x'):
   print('x pressed')
   calfunc = calibrate()

if key == ord('f'):
   camera.exposure_mode='off'



if calibflag:
   maxvect = calibvals[-1]-calibvals[0]
   #print(maxvect)
   currentvect = np.array(ave)-calibvals[0]
   projvect = onedvectproj(currentvect, maxvect)
   #print(projvect)
   #print('camera analog gain = : ' + str(camera.analog_gain))
   #print('camera digital gain = : ' + str(camera.digital_gain))




if key == ord('s'):
   if calibflag:
        conc = calfunc(projvect)
        sampleconcs.append(conc)
        sampleprojs.append(projvect)
        samplergbs.append(ave)
        print('current projection = ' + str(projvect))
        print('current conc = ' + str(conc))

xa = np.linspace(0, 1, 400)
plt.scatter(calibprojs, calibconcs)
plt.xlabel('calibprojs')
plt.ylabel('calibconcs')
plt.plot(xa, calfunc(xa))
plt.show()

plt.scatter(calibconcs, calibprojs)
plt.xlabel('calibconcs')
plt.ylabel('calibprojs')
plt.plot(calfunc(xa), xa)
plt.show()

sampleconcs = np.array(sampleconcs)
plt.scatter(np.arange(len(sampleconcs)), sampleconcs)
plt.xlabel('sample number')
plt.ylabel('estimated conc.')
plt.show()

i = 0
while os.path.exists('test%s.txt' % i):
   i += 1

np.savetxt('test%s.txt' % i, (np.array([0,0,0,0]), 'samplergbs', np.array(samplergbs), 'calibvals',
                calibvals, 'calibprojs',
                calibprojs, 'sampleprojs', sampleprojs, 'sampleconcs', sampleconcs), fmt='%s',
                header = 'samplergbs')

np.savez('test%s.npz' % i, calibvals=calibvals, calibprojs=calibprojs,
        samplergbs=np.array(samplergbs), sampleprojs=np.array(sampleprojs),
sampleconcs=sampleconcs)
```