THE EMERALD HEIGHTS INTERNATIONAL SCHOOL

THE CARBON DETECTOR

PROJECT REPORT

SESSION 2019-20

SUBJECT: INFORMATICS PRACTICES

PREPARED BY:
NARAYAN PHARKYA (XII-A)

APPROVED BY:
MR. ASHIRWAD MAHALKARI
(INTERNAL)

CERTIFICATE

THIS IS TO CERTIFY THAT **NARAYAN PHARKYA** OF CLASS **XII A** HAS SUCCESSFULLY COMPLETED A PROJECT TITLED **"THE CARBON DETECTOR"** OF THE SUBJECT **INFORMATICS PRACTICES** IN THE **ACADEMIC YEAR 2019-20.**

PROJECT INCHARGE
[MR. ASHIRWAD MAHALKARI]

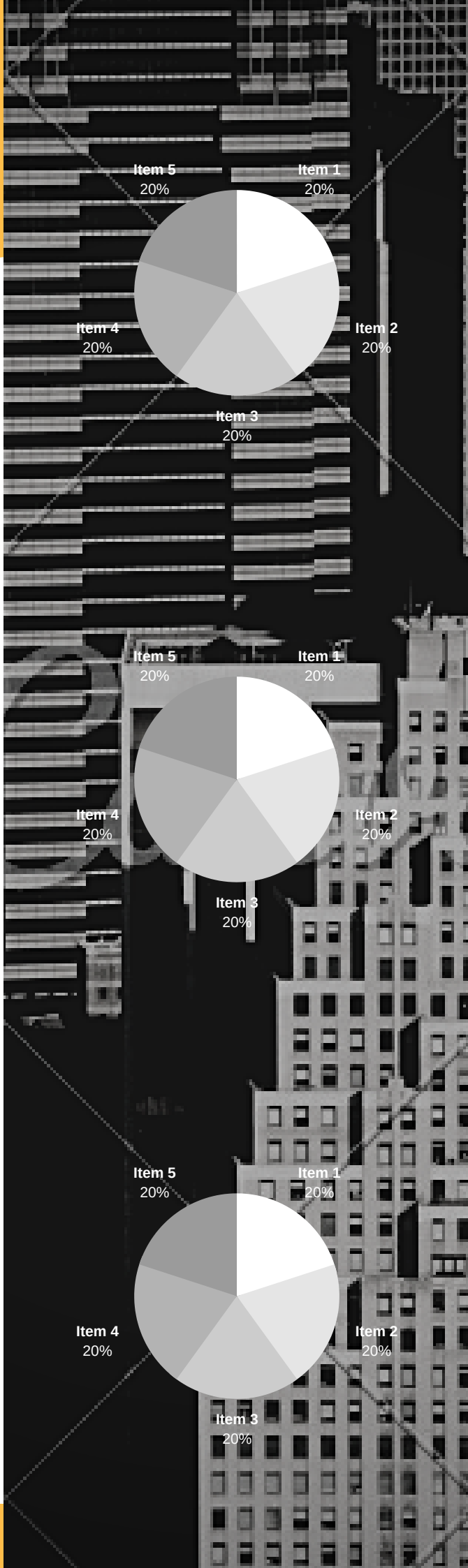
PRINCIPAL
[MR. SIDDHARTH SINGH]

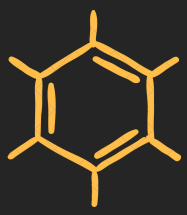
EXTERNAL'S SIGNATURE

ACKNOWLEDGEMENT

I AM EXTREMELY THANKFUL TO OUR PRESIDENT **MR. MUKTESH SINGH** AND OUR PRINCIPAL **MR.SIDDHARTH SINGH** FOR PROVIDING ME AN OPPORTUNITY TO WORK ON PROJECT TITLED**"THE CARBON DETECTOR"** AND FOR GIVING ME THEIR BLESSINGS AND ENCOURAGEMENT.IT IS A MATTER OF PRIDE AND PLEASURE TO EXPRESS MY WARM GRATITUDE TO MY TEACHER **MR.ASHIRWAD MAHALKARI** AND THE COMPUTER STAFF FOR KEEN INTEREST, UNCEASING RESISTANCE AND CRITICISM AT EVERY STAGE OF THIS PROJECT WORK . I AM ALSO THANKFUL TO THE C.B.S.E. FOR INCLUDING THE PROJECT WORK IN THE SYLLABUS.

TABLE OF CONTENT:





The Carbon Detector

ABSTRACT:

The Carbon Detector is not just a project but a combination of Hardware and Software that stands it out from other competing projects, it's a product designed by keeping in mind today's world scenario about climate change and how pollution is affecting our lives.If we talk about health issues today, we often hear about Heart attacks as the major cause of deaths, and deaths at a very early stage of life. The Project aim is to provide users with the information about the Air Quality,Humidity,Temperature and presence of Alcohol in their surroundings by plotting live graphs, in order to analyse the data easily. The user can also check the previous data of any particular time and date, which can help him understand and compare between data of different dates, the best part of the Project is that the user can get the report mailed to themselves and share it with whom they want to.

The back end of project is MySQL database where all information related to all four categories, i.e. Air Quality,Temperature,Humidity and Alcohol Detector date is stored and can be retrieved at any point of time.The project uses database connectivity where details of entered by the users are stored.The front end which is the user interface is the tkinter GUI form designed in Pycharm.This is done with the help of various GUI controls such as tkButton, tkLabel, tkText,tkEntry, and tkMenuBar. Also pictures through PhotoImage are inserted to add more creativity and pacify the interest of the users, this was all about the software part, coming to the Hardware part of our project, we have used Arduino UNO,as we found it to be most suitable for our project, and also to detect or check the quality of Air & Alcohol we have interface the Arduino with MQ-135, for Humidity & Temperature we have interface the Arduino with DHT-11.It provides a complete picture of an efficiently working software that can be incorporated and used by users in their daily life for carbon detection.

THE CARBON DETECTOR'S MISSION

The Carbon Detector is not only just a vision but a mission to see this world with zero carbon emission, a mission to brighten the world without carbon emission, a mission to enlighten the world with smiles on people's faces, a mission to create a world of our dreams!

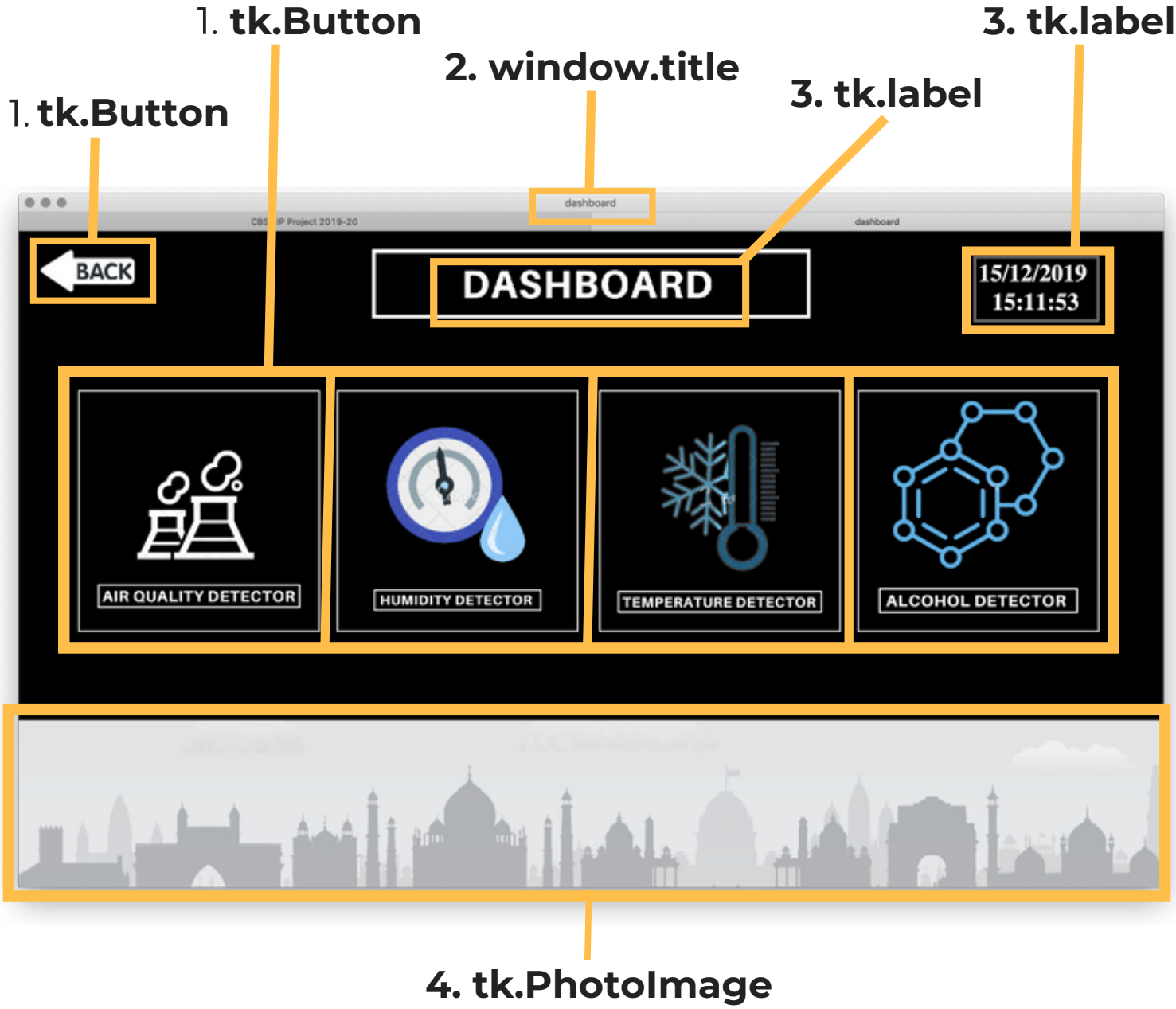
THE CARBON DETECTOR

DESIGN:

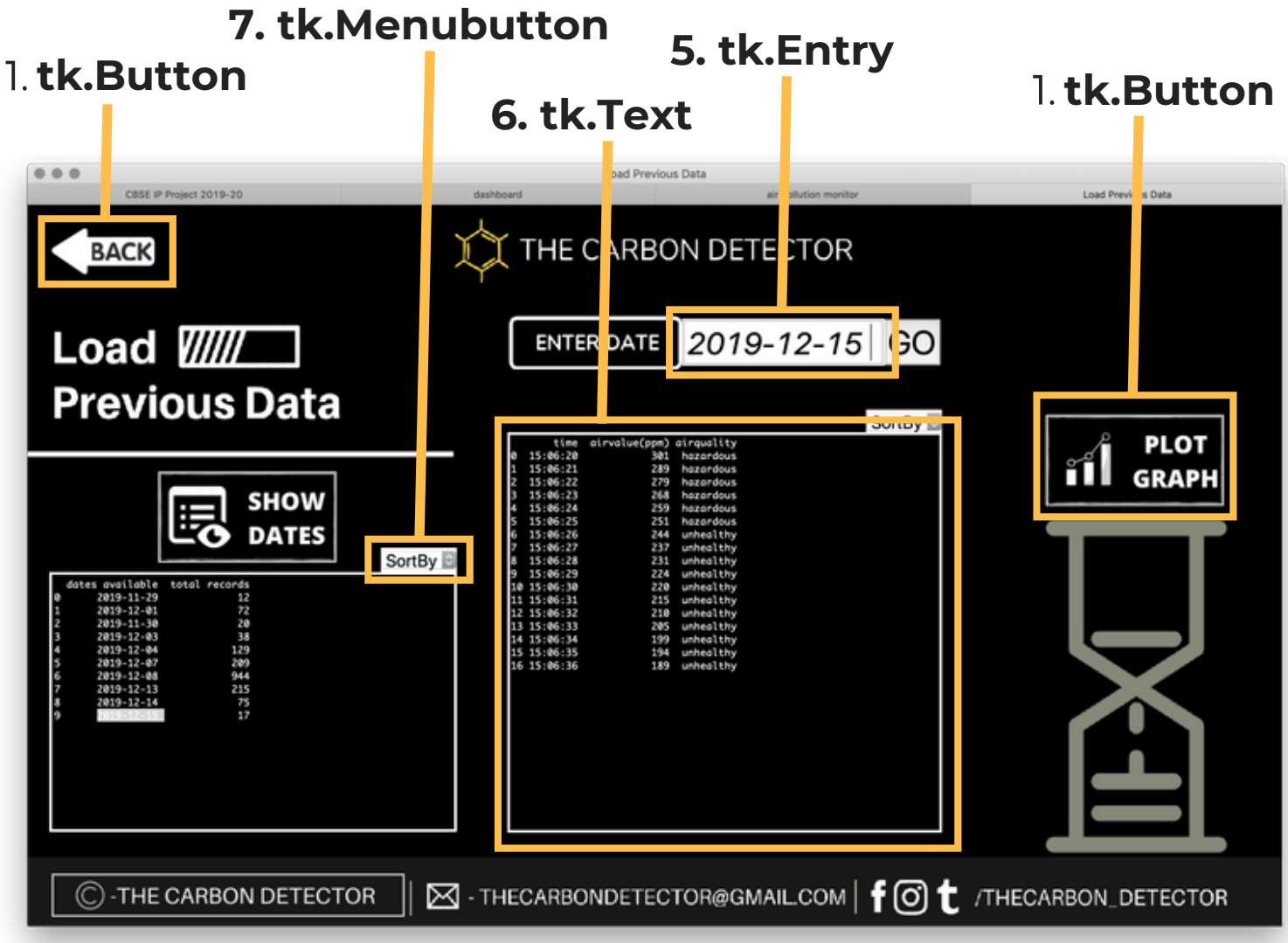
- We have used **TKINTER GUI** to show different windows and frames for a better user interface.
- We have used **MATPLOTLIB** to plot different graphs and show their figure windows on screen.
- We have used **PHOTOIMAGE** to insert different images in our project to make project realistic.
- We have used Tkinter **BUTTONS** , **ENTRY** to make our project user friendly and take user input.
- We have used Tkinter **TEXT** , **MENUBAR** which facilitates the user to see description and panel control in our window
- We have used **MYSQL.CONNECTOR** to connect to **MySQL** database and store the input data accordingly .
- We have used **TIME,DATETIME** module to show current date and time on window.
- We have use **SERIAL** module to connect to pc's serial port and access live values with help of sensors

FRONT END:

It is the user interface from where the data is collected, here we have used tkinter GUI to make it attractive and user friendly.The data entered here will then be processed in the MYSQL software as back end of the Project. Here are some functions that we have used in the front end:



1. **tk.Button:** The Button widget is used to add buttons in a tkinter window. These buttons can display text or images .
2. **window.title:** The window title function is used to rename the window of the software.
3. **tk.label:** This widget implements a display box where you can place text or images.
4. **tk.PhotoImage:** The function is use to add images to python and upload the image on a window or a button .
5. **tk.Entry:** The Entry widget is used to accept single-line text strings from a user.
6. **tk.Text:** Text widgets provide advanced capabilities that allow you to edit a multi-line text and format the way it has to be displayed, such as changing its color and font.
7. **tk.Menubutton:** A menubutton is the part of a drop-down menu. menubutton is associated with a Menu widget that can display the choices



BACK END:

It is the database where the data is actually stored .Data is mainly stored in DBMS (MySQL).
Database name: project
This was created and used by the use of the following MySQL query –
create database project;
use project;

The database contains two tables which can be seen by the the following mysql query-

show tables;

```
mysql> show tables;
+---------------------+
| Tables_in_project |
+---------------------+
| air                 |
| alcohol             |
+---------------------+
2 rows in set (0.01 sec)
```

TABLE AIR:

This table contains records of the air quality.The table stores date in **(yyyy-mm-dd)** format , time in **(hh-mm-ss)** and the air value in **ppm** associated with the time .The table also contain quality column to display air quality based on air value.

```
create table air
(
airvalue int(4),
time time ,
date date ,
quality varchar(10)
);
```

```
mysql> desc air;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| airvalue | int(4)      | YES  |     | NULL    |       |
| time     | time        | YES  |     | NULL    |       |
| date     | date        | YES  |     | NULL    |       |
| quality  | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

TABLE ALCOHOL:

This table contains records of the alcohol presence in surrounding. the table stores date in **(yyyy-mm-dd)** format , time in **(hh-mm-ss)** and the alcohol presence value associated with the time .The table also contain alcohol presence column to display alcohol presence based on alcohol value.

```
create table alcohol
(
alcoholvalue int(4),
time time ,
date date ,
alcoholpresence varchar(20)
);
```

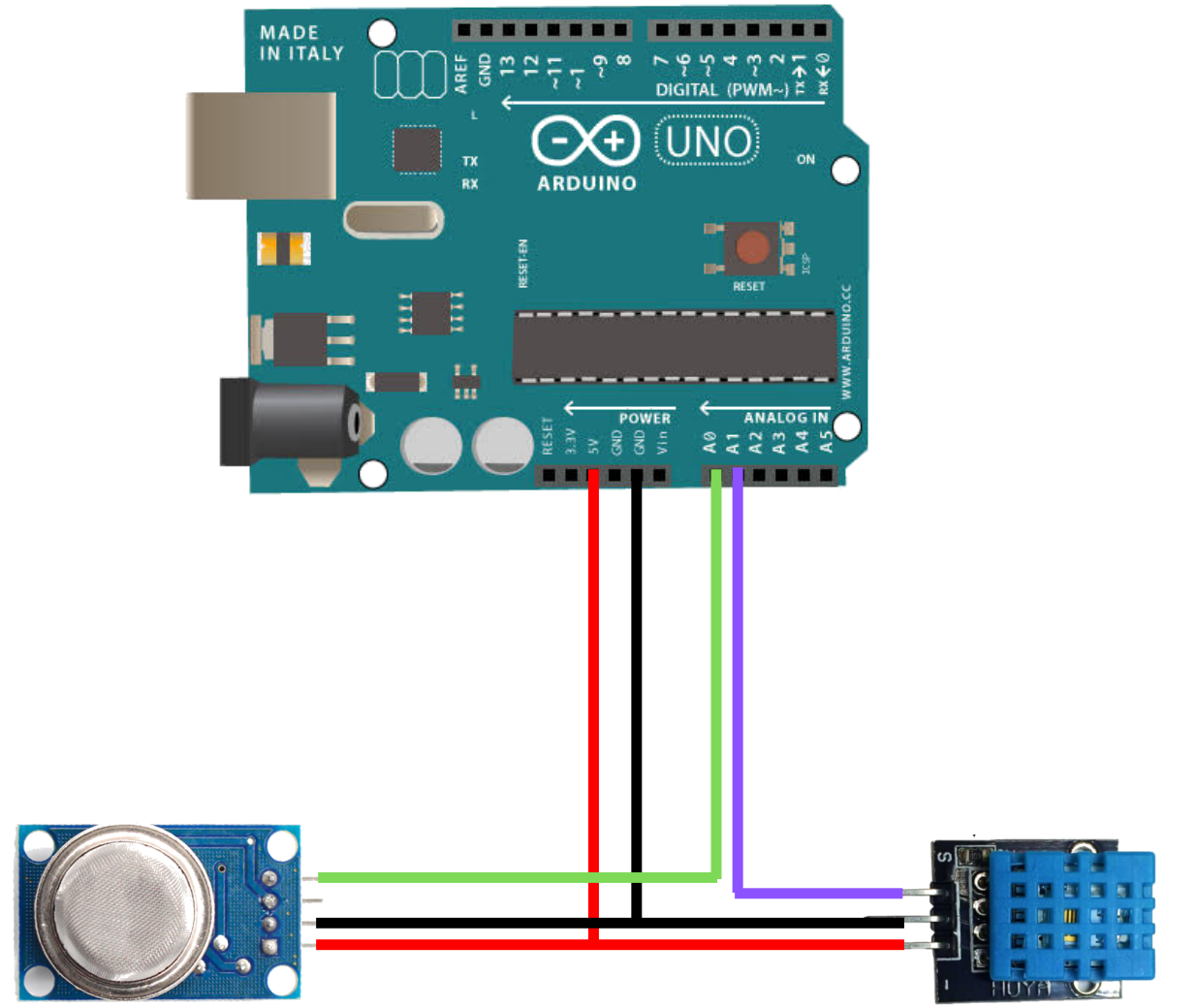
```
mysql> desc alcohol;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| alcoholvalue    | int(4)      | YES  |     | NULL    |       |
| time            | time        | YES  |     | NULL    |       |
| date            | date        | YES  |     | NULL    |       |
| alcoholpresence | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

ARDUINO UNO:

It is an open source community software that was most suitable for our project . here, the arduino code is used to initialize the **arduino UNO** board and **MQ- 135** sensor for detecting air quality and alcohol presence & **DHT-11** sensors for detecting temperature and humidity.

Here is the **circuit diagram** for the same:

ARDUINO UNO



[CIRCUIT DIAGRAM]

ARDUINO UNO:

Here's the code that we wrote over the Arduino:

mq135 §

```
int val = 0;


void setup(){
Serial.begin(9600); // sets the serial port to 9600
 }
void loop()
{
  val = analogRead(0);
  Serial.println(val);
  delay(1000); // wait 1000ms for next reading
}
```

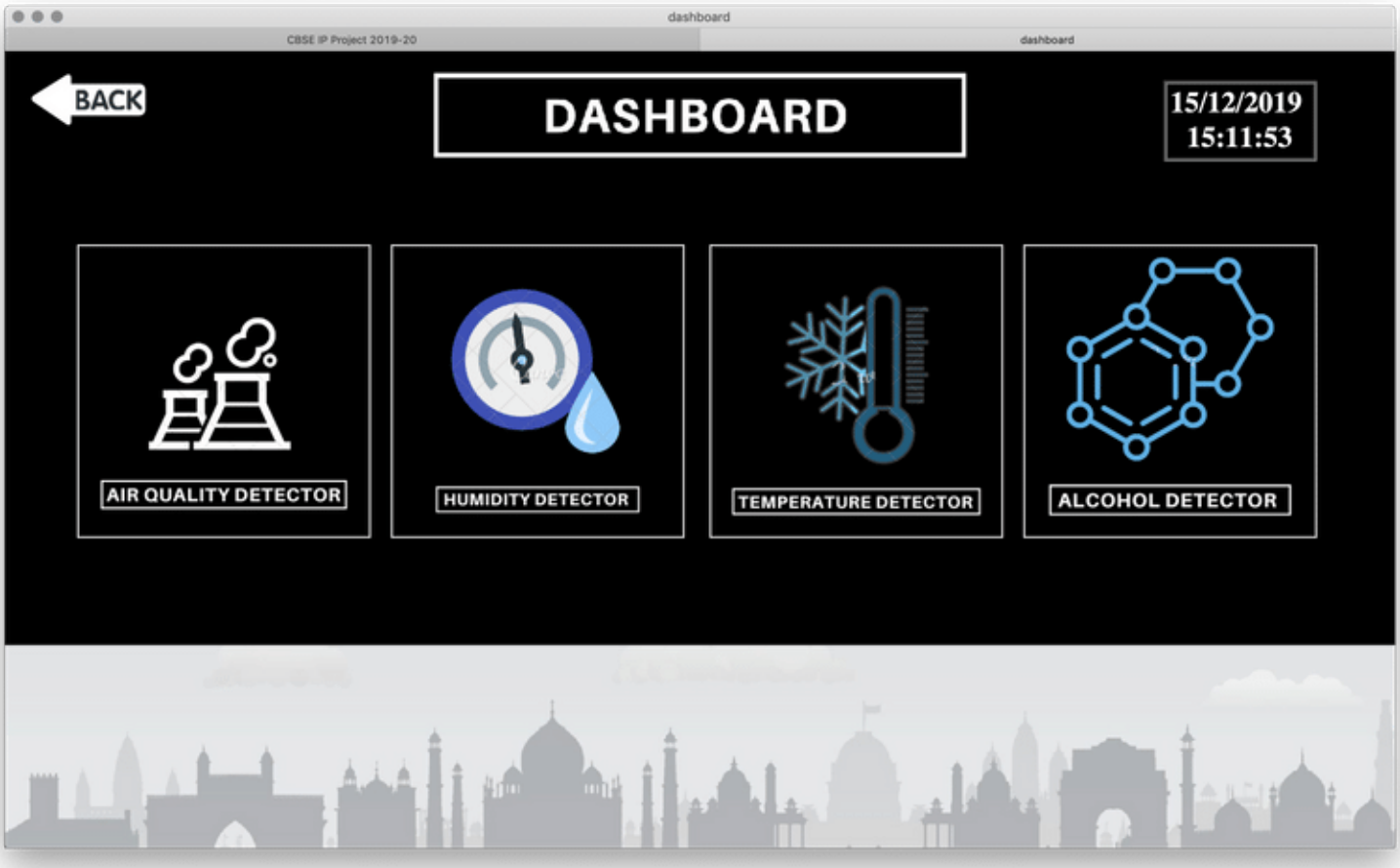
SOURCE CODE:

CODING FOR THE PARENT WINDOW AND IMPORTED MODULES:


```
import tkinter as tk
import pandas
from PIL import Image, ImageTk
import time
import datetime as dt
import serial
import mysql.connector as connector
import matplotlib.pyplot as plt
import matplotlib.animation as anm
import smtplib
import ssl
```

```
ser = serial.Serial("/dev/cu.usbmodem14101", 9600)
window = tk.Tk()
window.title('CBSE IP Project 2019-20')
window.geometry('1440x850')
window.configure(bg='black')
load = Image.open("bcg1.png")
load = load.resize((1440, 850))
bcgimg = ImageTk.PhotoImage(load)
img = tk.Label(window, bg="black", image=bcgimg, )
img.pack()
..
..
..
..
..
..
..
..
..
..
..
..
..
..
..
..
window.bind('<Button-1>', dashboard)
window.bind('<space>', dashboard)
window.bind('<Return>', dashboard)
window.mainloop()
```

CODING FOR THE SECOND WINDOW:




```
def dashboard(event):
    homedash = tk.Toplevel(window)
    homedash.title("dashboard")
    homedash.geometry("1440x850")
    homedash.configure(bg="black")
    load1 = Image.open("bcg.png")
    load1 = load1.resize((1440, 850))
    gifbcg = ImageTk.PhotoImage(load1)
    img1 = tk.Label(homedash, bg="white", image=gifbcg, )
    img1.image = gifbcg
    img1.place(x=-5, y=-5)
    tk.Label(homedash, bg="white", image=gifbcg).pack()
    clock = tk.Label(homedash, font=("times", 30, 'bold'),      bg="b" ,
                              fg="white", borderwidth=4, relief="groove")
    clock.place(x=1200, y=30)
    def tick(time1=''):
        time2 = time.strftime("%d/%m/%Y \n%H:%M:%S")
```

```
        if time2 != time1:
            time1 = time2
            clock.config(text=time2)
        clock.after(200, tick)
..
..
..
..
..

    air = tk.PhotoImage(file="air.png")
    airbutton = tk.Button(homedash, text="air quality monitor",
                                image=air, width=300, height=300,
                                command=airquality)
    airbutton.image = air
    airbutton.config(bg='black', fg="black", bd=0)
    airbutton.place(x=75, y=200)

    hum = tk.PhotoImage(file="hum.png")
    humidbutton = tk.Button(homedash, text="humidity monitor",
                                    image=hum, width=300, height=300,
                                    command=humidity)
    humidbutton.image = hum
    humidbutton.config(bg='black', fg="black", bd=0)
    humidbutton.place(x=400, y=200)

    temp = tk.PhotoImage(file="temp.png")
    tempbutton = tk.Button(homedash, text="temperature
                                monitor", image=temp, width=300,
                                height=300, command=temperature)
    tempbutton.image = temp
    tempbutton.config(bg='black', fg="black", bd=0)
    tempbutton.place(x=730, y=200)  # isse position set ki
```

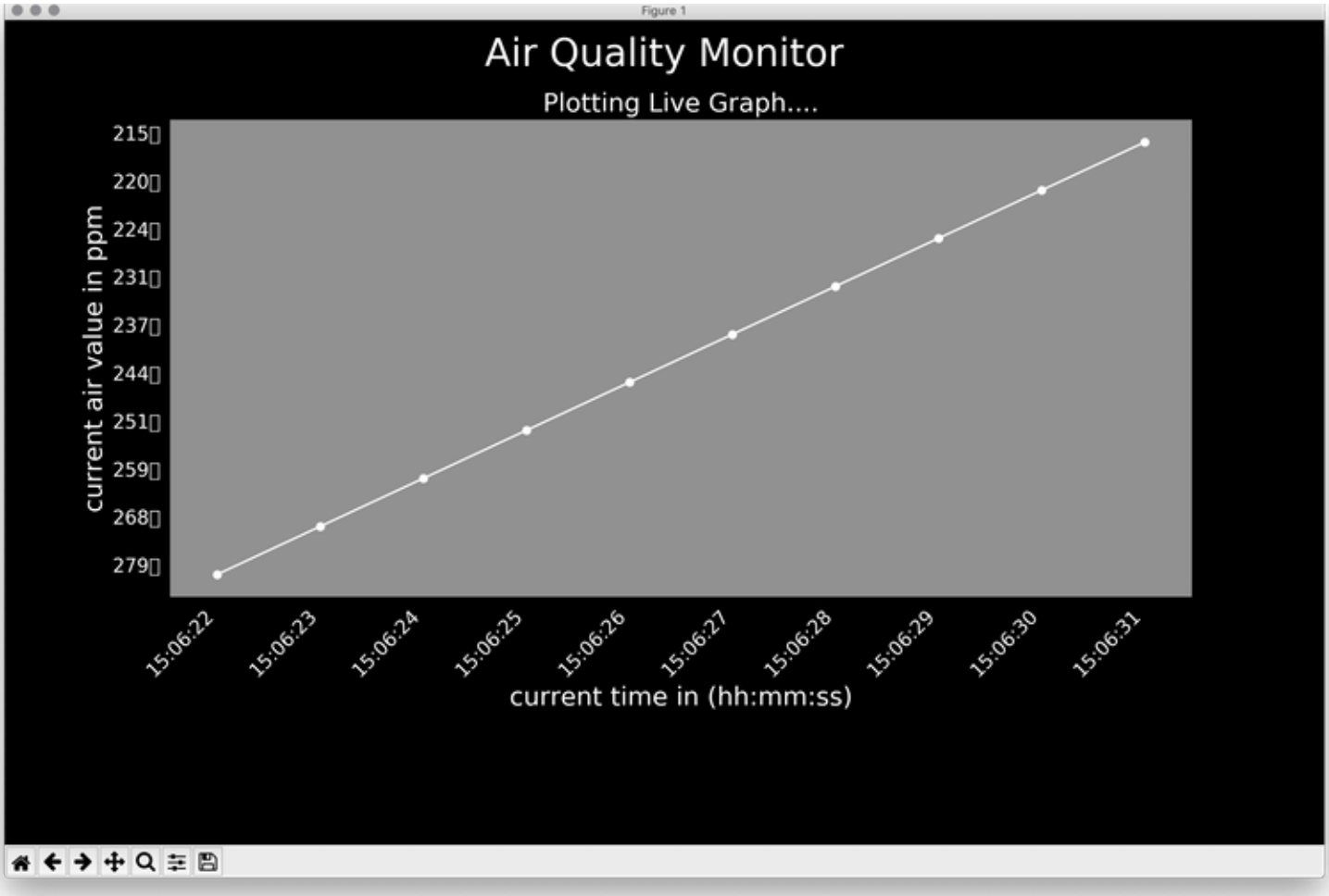
```
 alco = tk.PhotoImage(file="alco.png")
 alcoholbutton = tk.Button(homedash, text="alcohol monitor",
                                     image=alco, width=300, height=300,
                                     command=alcohol)
alcoholbutton.image = alco
alcoholbutton.config(bg='black', fg="black", bd=0)
alcoholbutton.place(x=1055, y=200)

back = tk.PhotoImage(file="back.png")
backbutton = tk.Button(homedash, text="back", image=back,
                                  width=150, height=80,
                                  command=homedash.destroy)
backbutton.image = back
backbutton.config(bg='black', fg="black", bd=0)
backbutton.place(x=10, y=10)


tick()
```

CODING FOR THE AIR QUALITY WINDOW:


```
def airquality():
    airwindow = tk.Toplevel(homedash)
    airwindow.title("air pollution monitor")
    airwindow.geometry("1440x850")
    airwindow.configure(bg='black')
    load2 = Image.open("airb.png")
    load2 = load2.resize((1440, 850))
    airb = ImageTk.PhotoImage(load2)
    img2 = tk.Label(airwindow, bg="white", image=airb, )
    img2.image = airb
    img2.place(x=-3, y=-3)
    ..
    ..
    ..
    ..
    ..
    ..
```

```
 live = tk.PhotoImage(file="live.png")
 livegraphbutton = tk.Button(airwindow, text="live graph
                                plotting", image=live,  width=300,
                                    height=200,   command=lplot)
livegraphbutton.image = live
livegraphbutton.config(bg='black', fg="black")
livegraphbutton.place(x=1075, y=110)
load = tk.PhotoImage(file="load.png")
loadgraphbutton = tk.Button(airwindow, text="load graph",
                                      image=load, width=300,
                                      height=200, command=loadp)
loadgraphbutton.image = load
loadgraphbutton.config(bg='black', fg="black")
loadgraphbutton.place(x=1075, y=330)
mail = tk.PhotoImage(file="mail.png")
mailgraphbutton = tk.Button(airwindow, text="mail graph",
                                      image=mail, width=300,
                                      height=200, command=mailr)
mailgraphbutton.image = mail
mailgraphbutton.config(bg='black', fg="black")
mailgraphbutton.place(x=1075, y=550)
cri = tk.PhotoImage(file="cr.png")
crbutton = tk.Button(airwindow, text="back", image=cri,
                           width=400, height=45, command=cr)
crbutton.image = cri
crbutton.config(bg='black', fg="black", bd=0)
crbutton.place(x=28, y=775)
back = tk.PhotoImage(file="back.png")
backbutton = tk.Button(airwindow, text="back",
                                image=back,  width=150, height=80,
                                command=airwindow.destroy)
backbutton.image = back
backbutton.config(bg='black', fg="black", bd=0)
backbutton.place(x=10, y=10)
```

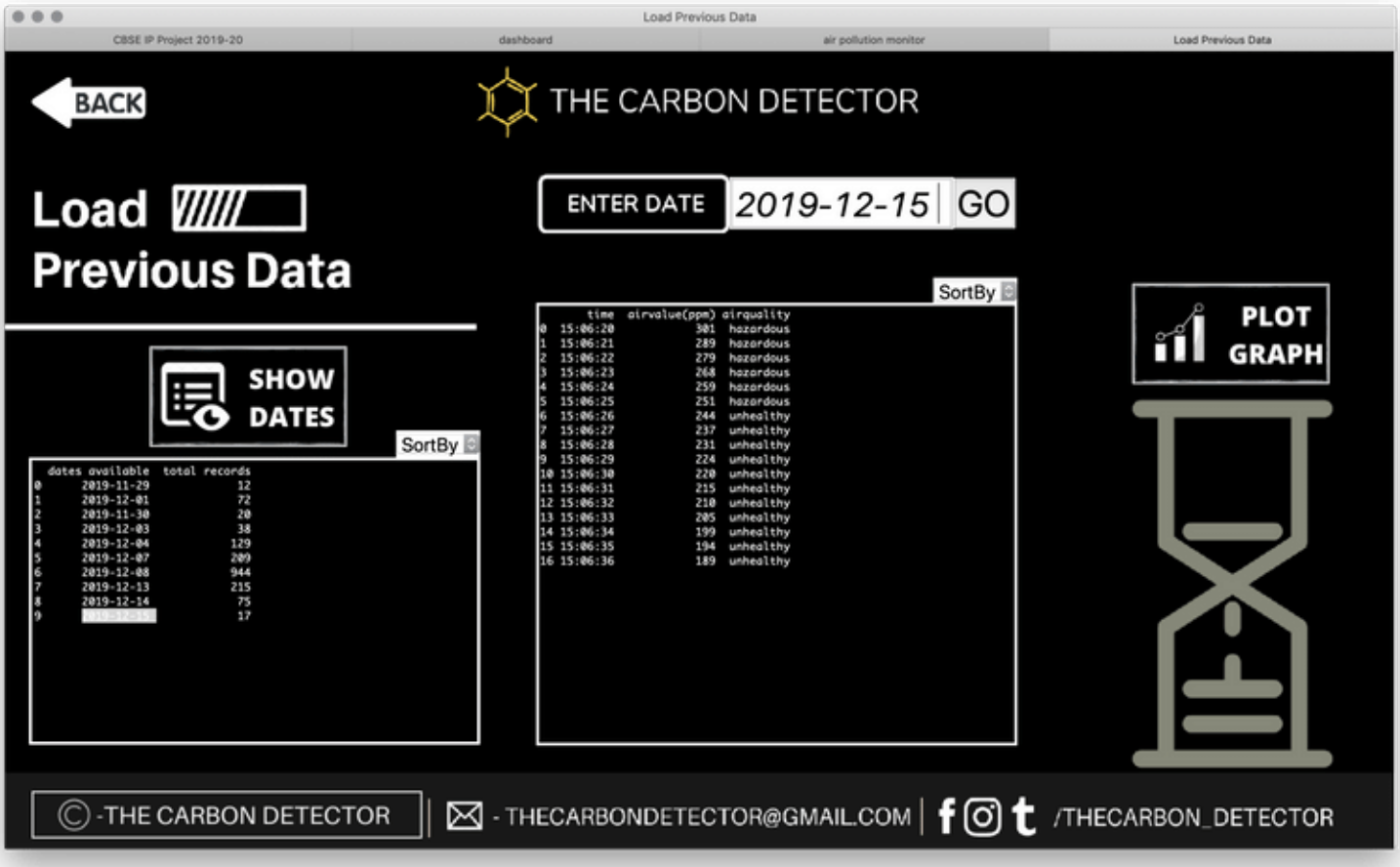
CODING FOR THE LIVE PLOTTING WINDOW:




```
def lplot():
        fig = plt.figure(figsize=(16, 9), facecolor='black',
                                          edgecolor="white")
        ax = fig.add_subplot(1, 1, 1)
        ax.set_facecolor("grey")
        xs = []
        ys = []
        def contiplot(i, xs, ys):
            b = ser.readline()
            c = int(b)
           tim = dt.datetime.now().strftime("%H:%M:%S")
           dat = dt.datetime.now().strftime("%Y/%m/%d")
           xs.append(tim)
           ys.append(b)
           xs = xs[-10:]
           ys = ys[-10:]
```

```
        myconnection = connector.connect(host="localhost",
                                         user="root", password="qwert123456",
                                         database="project")
        mycursor = myconnection.cursor()
        mycursor.execute("insert into air values({},'{}','{}',NULL)"
                                        .format(c, tim, dat))
        mycursor.execute("update air set quality ='good'  where
                                         airvalue <= 50")
        mycursor.execute("update air set quality ='moderate'
                                        where airvalue between 51 and 150")
        mycursor.execute("update air set quality ='unhealthy'
                                       where airvalue between 151 and 250"
        mycursor.execute("update air set quality ='hazardous'
                                        where airvalue >250")
        myconnection.commit()
        myconnection.close()
        ax.clear()
        ax.plot(xs, ys, c="white")
        ax.scatter(xs, ys, c="white")
        plt.yticks(c='w',fontsize =15)
        plt.xticks(rotation=45, ha="right", c='w',fontsize=15)
        plt.xlabel("current time (hh:mm:ss)",c="w",fontsize=20)
        plt.ylabel("current air value ppm",c="w",fontsize=20)
        plt.subplots_adjust(bottom=0.30)
        plt.title("Plotting Live Graph....",fontsize=20, c="w")
        plt.suptitle("Air Quality Monitor", fontsize=30, c="w")

ani = anm.FuncAnimation(fig, contiplot, fargs=(xs, ys),
                                        interval=1000)
plt.show()
```

CODING FOR THE PLOT PREVIOUS WINDOW:




```
def loadp():
    def loddata():
        airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
        mycursor = airdb.cursor()
        mycursor.execute('select date,count(*)"number of record"
                         from air group by date')
        result = mycursor.fetchall()
        pandas.set_option('max_rows', None)
        DF = pandas.DataFrame(result, columns=['dates available',
                                               'total records'])
        txtdata.delete("1.0", tk.END)
        txtdata.insert(tk.INSERT, DF)
        txtdata.config()
```

```
def dateasec():
    airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
    mycursor = airdb.cursor()
    mycursor.execute('select date,count(*)"number of record"
                        from air  group by date  order by date')
    result = mycursor.fetchall()
    pandas.set_option('max_rows', None)
    DF = pandas.DataFrame(result, columns=['dates
                                available',  'total records'])
    txtdata.delete("1.0", tk.END)
    txtdata.insert(tk.INSERT, DF)
    txtdata.config()
def datedesc():
    airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
    mycursor = airdb.cursor()
    mycursor.execute('select date,count(*)"number of record"
                        from air  group by date  order by date desc')
    result = mycursor.fetchall()
    pandas.set_option('max_rows', None)
    DF = pandas.DataFrame(result, columns=['dates available',
                                'total records'])
    txtdata.delete("1.0", tk.END)
    txtdata.insert(tk.INSERT, DF)
    txtdata.config()
def norasec():
    airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
    mycursor = airdb.cursor()
    mycursor.execute('select date,count(*)"number of record"
                        from air  group by date  order by count(*) ')
    result = mycursor.fetchall()
    pandas.set_option('max_rows', None)
```

```
        DF = pandas.DataFrame(result, columns=['dates available',
                                               'total records'])
        txtdata.delete("1.0", tk.END)
        txtdata.insert(tk.INSERT, DF)
        txtdata.config()
loadwindow = tk.Toplevel(airwindow)
loadwindow.title("Load Previous Data")
loadwindow.geometry("1440x850")
load1 = Image.open("loadscr.png")
loadscr = ImageTk.PhotoImage(load1)
img1 = tk.Label(loadwindow, bg="white", image=loadscr, )
img1.image = loadscr
txtdata = tk.Text(loadwindow, width=65, height=19,
                  fg="white", bg="black", borderwidth="2")
txtdata.place(x=25, y=420)
sortbutton = tk.Menubutton(loadwindow, text="SortBy",
                                       font="helvica 20")
sortbutton.menu = tk.Menu(sortbutton, tearoff=0)
sortbutton["menu"] = sortbutton.menu
sortbutton.menu.add_command(label="date ascending",
                                       command=dateasec)
sortbutton.menu.add_command(label="date descending",
                                       command=datedesc)
sortbutton.menu.add_command(label="total records
                                  ascending", command=norasec)
sortbutton.menu.add_command(label="total records
                                 descending", command=nordesc)
sortbutton.place(x=405, y=392)
load = tk.PhotoImage(file="dates.png")
loadbutton = tk.Button(loadwindow, text='Show Dates',
                                  image=load, width=200, height=100,
                           command=loddata,   font='times 20 bold')
loadbutton.image = load
loadbutton.place(x=150, y=305)
```

```
def extractdata():
    enterdate = inputenterd.get()
    airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
    mycursor = airdb.cursor()
    mycursor.execute("select time , airvalue,quality from air
                where date = '{}'".format(enterdate))
    result = mycursor.fetchall()
    pandas.set_option('max_rows', None)
    DF = pandas.DataFrame(result, columns=['time', '
                airvalue(ppm)', 'airquality'])
    txtedata.delete("1.0", tk.END)
    txtedata.insert(tk.INSERT, DF)
    txtedata.config()
    t = DF["time"].tolist()
    a = DF['airvalue(ppm)'].tolist()
    final_t = list()
    for x in t:
        x = str(x)
        s = x[7:]
        final_t.append(s)
  plt.plot(final_t, a)
def timeasec():
    enterdate = inputenterd.get()
    airdb = connector.connect(host="localhost", user="root",
                passwd="qwert123456", database="project")
    mycursor = airdb.cursor()
    mycursor.execute( "select time  , airvalue,quality from air
            where date = '{}' order by time".forma t(enterdate))
    result = mycursor.fetchall()
    pandas.set_option('max_rows', None)
    DF = pandas.DataFrame(result, columns=['time',
                'airvalue(ppm)', 'airquality'])
    txtedata.delete("1.0", tk.END)
```

```
        txtedata.insert(tk.INSERT, DF)
        txtedata.config()
        t = DF["time"].tolist()
        a = DF['airvalue(ppm)'].tolist()
        final_t = list()
        for x in t:
            x = str(x)
            s = x[7:]
            final_t.append(s)
        plt.plot(final_t, a)
    def plotp():
        plt.xticks(rotation=45, ha="right")
        plt.subplots_adjust(bottom=0.30)
        plt.show()

plp = tk.PhotoImage(file="graphbutton.png")
plpbutton = tk.Button(loadwindow, text='plot graph',
                      image=plp, width=200, height=100, command=plotp,
                      font='times 20 bold', bd=0)
plpbutton.image = plp
plpbutton.place(x=1167, y=240)
sortbutton = tk.Menubutton(loadwindow, text="SortBy",
                                                font="helvica 20")
sortbutton.menu = tk.Menu(sortbutton, tearoff=0)
sortbutton["menu"] = sortbutton.menu
sortbutton.menu.add_command(label="time ascending",
                                                command=timeasec)
sortbutton.menu.add_command(label="time descending",
                                                command=timedesc)
sortbutton.menu.add_command(label="airquality highest first",
                                                command=airhigh)
sortbutton.menu.add_command(label="airquality lowest first",
                                                 command=airlow)
sortbutton.place(x=961, y=234)
```

```
 inputenterd = tk.Entry(loadwindow, font="helvica 35 italic"
                                                    width=10)
inputenterd.place(x=750, y=130)
txtedata = tk.Text(loadwindow, width=70, height=30,
                          fg="white", bg="black", borderwidth="0")
txtedata.place(x=550, y=260)
go = tk.Button(loadwindow, text="GO", font="helvica 38",
                                              command=extractdata)
go.place(x=982, y=131)
back = tk.PhotoImage(file="back.png")
backbutton = tk.Button(loadwindow, text="back", image=back,
          width=150, height=85, command=loadwindow.destroy)
backbutton.image = back
backbutton.config(bg='black', fg="black", bd=0)
backbutton.place(x=10, y=10)
def cr():
      crwindow = tk.Toplevel(loadwindow)
      crwindow.title("copyrights")
      crwindow.configure(bg='black')
      load3 = Image.open("copyrights.png")
      load3 = load3.resize((1440, 850))
      humb = ImageTk.PhotoImage(load3)
      img3 = tk.Label(crwindow, bg="white", image=humb, )
      img3.image = humb
      img3.place(x=-3, y=-3)
back = tk.PhotoImage(file="back.png")
backbutton = tk.Button(crwindow, text="back", image=back,
              width=150, height=80,  command=crwindow.destroy)
backbutton.image = back
backbutton.config(bg='black', fg="black", bd=0)
backbutton.place(x=80, y=1)
```

```
cri = tk.PhotoImage(file="cr.png")
crbutton = tk.Button(loadwindow, text="back", image=cri,
                                  width=400, height=45, command=cr)
crbutton.image = cri
crbutton.config(bg='black', fg="black", bd=0)
crbutton.place(x=28, y=765)
```

CODING FOR THE MAIL WINDOW:


```
def mailr():
    mailwindow = tk.Toplevel(airwindow)
    mailwindow.title("mail report")
    mailwindow.geometry("1440x850")
    mailwindow.configure(bg='black')
    tk.Label(mailwindow, text="Mail Report", bg="black",
                                fg="white", font='times 44 bold
                                        ').place(x=600, y=10)
```

```
load4 = Image.open("mailb.png")
load4 = load4.resize((1440, 850))
mailb = ImageTk.PhotoImage(load4)
img4 = tk.Label(mailwindow, bg="white", image=mailb, )
img4.image = mailb
img4.place(x=-3, y=-3)
port = 465
smtp_server = 'smtp.gmail.com'
sender = 'carbondetector@gmail.com'
password = "qwert1234566"
recievermailid = tk.Entry(mailwindow, font="helvica 45",
                                    bg="black", fg="white", bd=2)
recievermailid.place(x=340, y=624)
reciever = recievermailid
message = """\
subject: trial mail
hi this is thecarbondetector"""
def sendmail():
    context = ssl.create_default_context()
            with smtplib.SMTP_SSL(smtp_server, port,
                                  context=context) as server:
                server.login(sender, password)
                server.sendmail(sender, reciever, message)
send = tk.PhotoImage(file="sendbutton.png")
sendbutton = tk.Button(mailwindow, text="back",
      image=send, width=65, height=61, command=sendmail)
sendbutton.image = send
sendbutton.config(bg='black', fg="black", bd=0)
sendbutton.place(x=910, y=624)
back = tk.PhotoImage(file="back.png")
backbutton = tk.Button(mailwindow, text="back",   image=
 back, width=150, height=85, command=mailwindow.destroy)
backbutton.image = back
backbutton.config(bg='black', fg="black", bd=0)
```

CODING FOR THE COPYRIGHTS WINDOW:


```
def cr():
    crwindow = tk.Toplevel(airwindow)
    crwindow.title("copyrights")
    crwindow.configure(bg='black')
    load3 = Image.open("copyrights.png")
    load3 = load3.resize((1440, 850))
    humb = ImageTk.PhotoImage(load3)
    img3 = tk.Label(crwindow, bg="white", image=humb, )
    img3.image = humb
    img3.place(x=-3, y=-3)
    back = tk.PhotoImage(file="back.png")
    backbutton = tk.Button(crwindow, text="back", image=back,
                width=150, height=80, command=crwindow.destroy)
    backbutton.image = back
    backbutton.config(bg='black', fg="black", bd=0)
    backbutton.place(x=80, y=1)
```

NOTE-

THE OTHER WINDOWS :

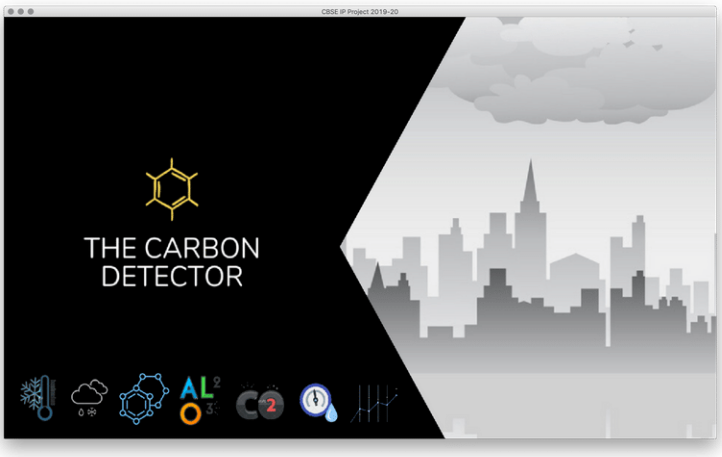
- **-HUMIDITY DETECTOR**
- **-TEMPERATURE DETECTOR**
- **-ALCOHOL DETECTOR**

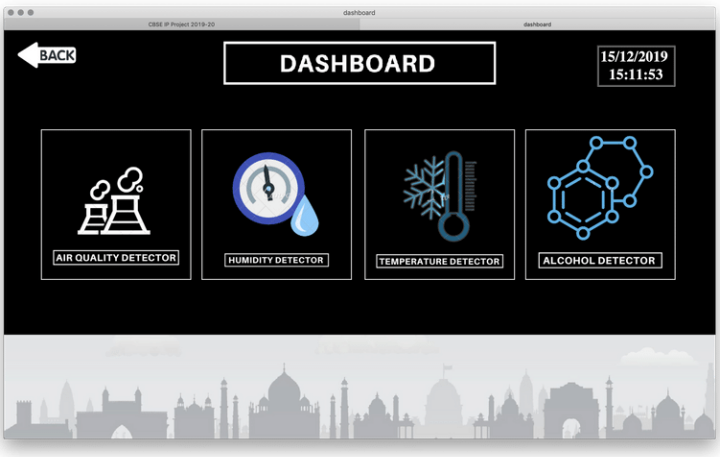
ARE ALSO HAVING THE SAME SOURCE CODE . TO SEE THE COMPLETE CODE KINDLY REFER TO THE LINK GIVE BELOW:

HTTPS://DRIVE.GOOGLE.COM/FILE/D/1XHQEWIECOOYTSJHWMZNHO5DUO8UWXG1Q/VIEW

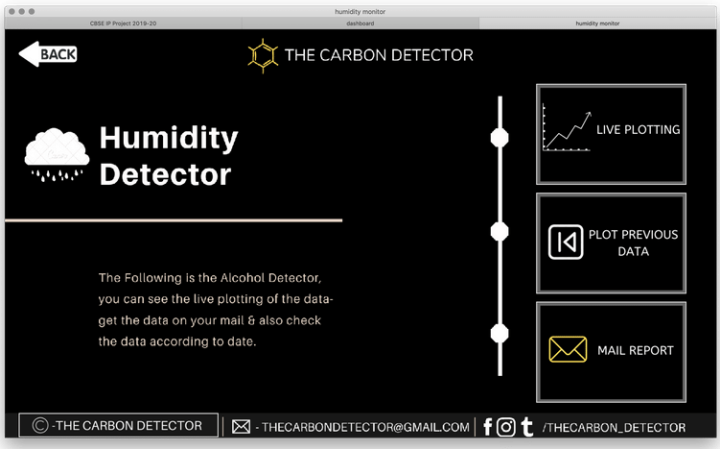
OR SCAN THE QR CODE

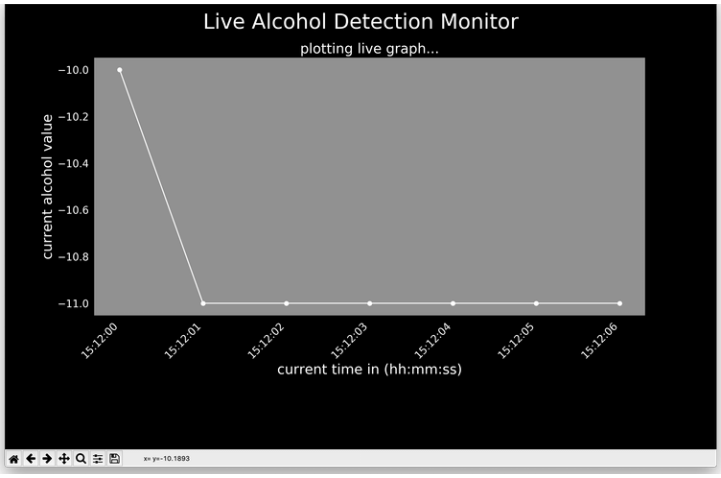
AND THIS IS WHAT IT LOOKS LIKE!

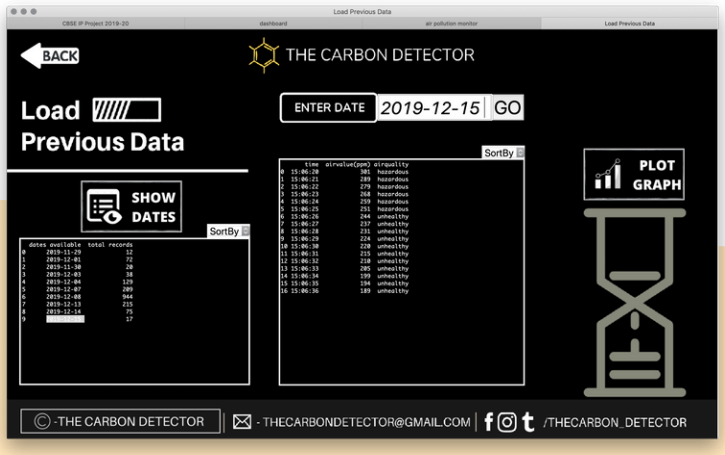




AND THIS IS WHAT IT LOOKS LIKE!

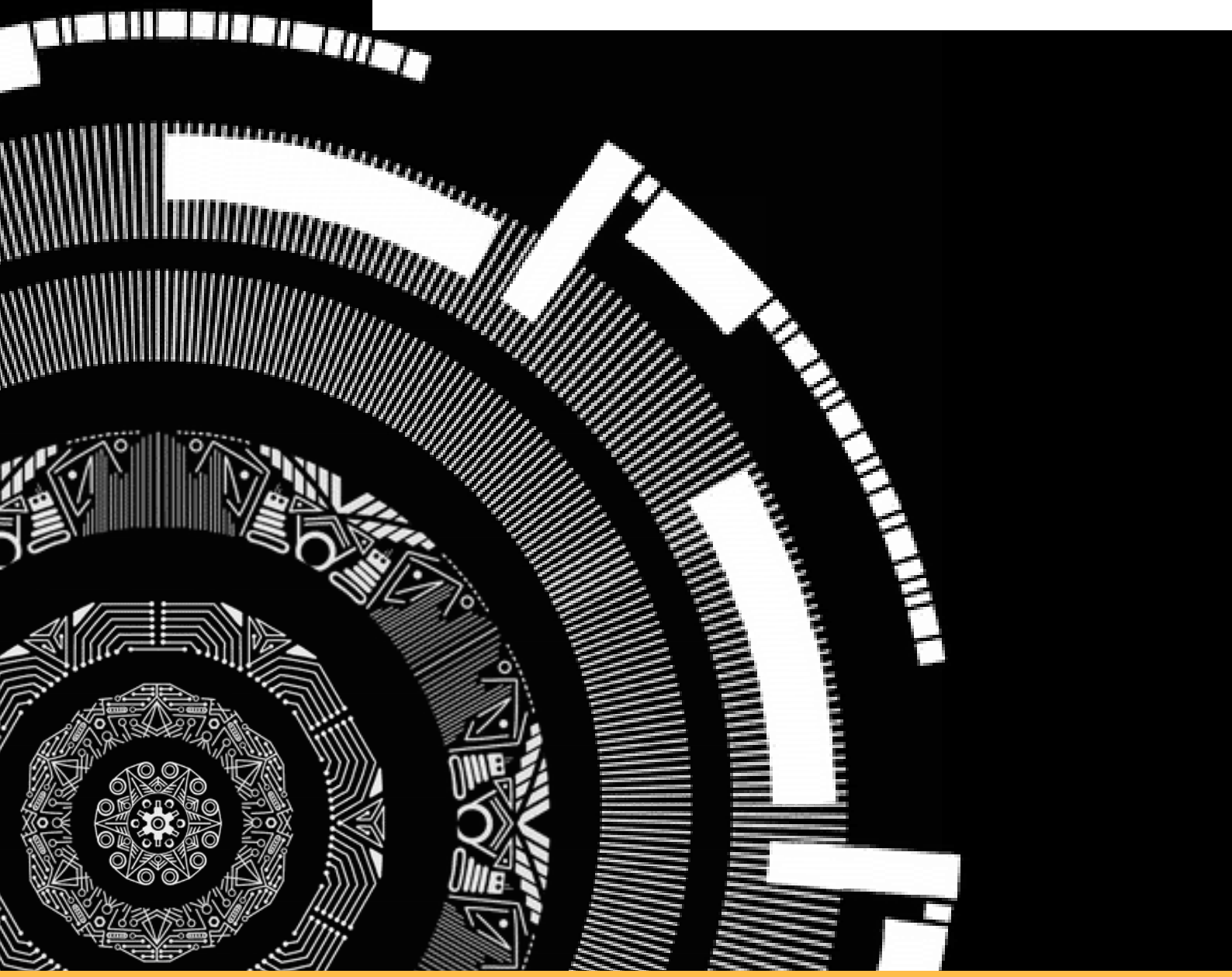


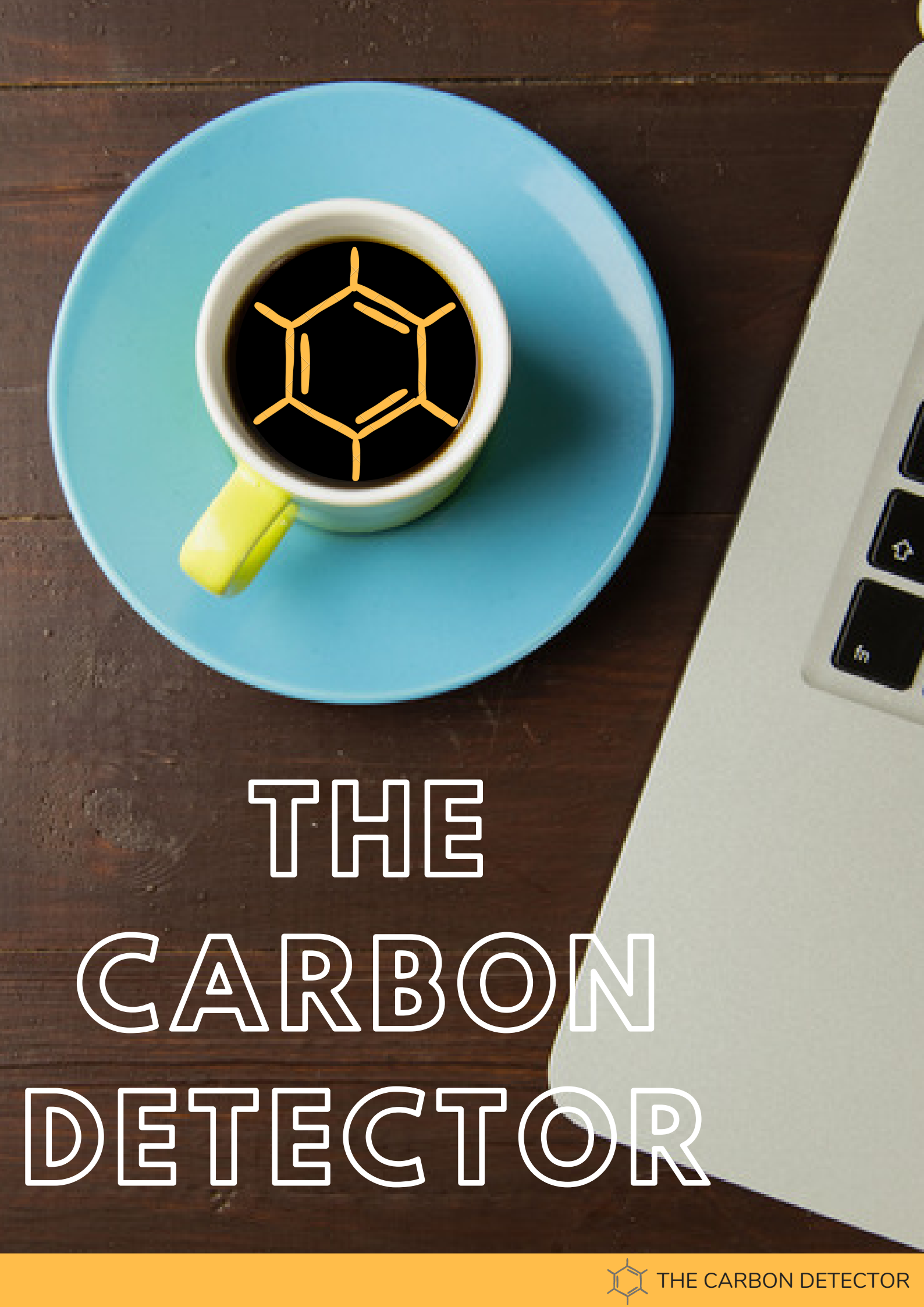




BIBLIOGRAPHY:

- https://www.geeksforgeeks.org/
- https://github.com/python
- https://www.arduino.cc/
- https://stackoverflow.com/
- https://mathwork.com/





THE CARBON DETECTOR