IoT Based Carbon Dioxide Density Sensing and Prediction System

Imran Chowdhury Dipto

Department of Electronics, Computing and Mathematics
University of Derby, UK
100519293@unimail.derby.ac.uk

Abstract. Carbon Dioxide Levels are rising drastically and it is one of the major concerns of the World currently. This is because Carbon Dioxide is responsible for global warming, health issues and so on. The data collection of concentration of Carbon Dioxide Levels in air and making predictions regarding the future levels is important so that authorities in cities, regions or states could take preventative actions to maintain acceptable levels of Carbon Dioxide in Air. Industries are adopting IoT at a rapid rate hence, IoT is selected for the task. The aim of this project is to build a prototype IoT based Carbon Dioxide Density Measuring and Prediction System and present its Technical and Business case. To reach the aim, a System is made using Arduino, MQ-135 and an LCD Display powered by a computer. Then data generated from the device was collected, analysed and a Machine Learning Model was trained on the data. To explain the Technical and Business case of the System a Framework has been used. From the results, it is found that the prototype system if deployed commercially would be beneficial for a business. Overall the system is easy to build, inexpensive due to the components used and all the software being open-source. The results generated were reliable and there are many future upgrade options available for the System.

Keywords: IoT · Carbon Dioxide · Arduino · MQ-135 · Prediction.

Table of Contents

1 Introduction

It is known that Carbon Dioxide is a colourless and odourless gas that is soluble in water and is seen as bobbles in fizzy drinks. However, it is also a greenhouse gas and a by-product of burned materials containing carbon and respiration (Lehto, 2018). Moderate to high levels of CO2 could cause headaches and fatigue also, higher concentrations could cause nausea, dizziness and vomiting. Loss of consciousness could result from extreme concentration levels. A safe level of CO2 in occupied spaces with adequate air exchange ranges between 350-1000 Parts Per Million (PPM) (Bonino, 2016).

To maintain adequate levels of Carbon Dioxide at places such as homes, offices, industries and so on an IoT based Carbon Dioxide Sensing System had been proposed previously and in this research, a complete demonstration of the project will be shown and data generated from the device will be utilised to display its prospects of being used in large scale.

2 Requirements for the Data Generation

2.1 Components

Arduino UNO-R3: To carry out the project the Arduino UNO-R3 is used. It is chosen because it is the most flexible hardware-based on ATmega328P which could be programmed according to the required function to be performed. It has 6 analogue inputs, 14 digital input/output pins (the 6 pins could be used as PWM outputs), a USB connection, a 16 MHz quartz crystal, SPI, serial interface, a reset button, a power jack and an ICSP header (Pal et al, 2017). The board is shown in Figure 1 of Appendix A.

MQ-135: Shown in Figure 2 in Appendix B the MQ-135 is used as the sensor for CO2. This sensor contains a sensitive material called SnO2. The conductivity of SnO2 is lower in clean air and it increases with a higher concentration of target gas to be sensed. The range of detection of the sensor is between 10-10,000 PPM with a voltage of around 5V +/-0.1 AC or DC (Kaur et al, 2016).

LCD Advanced Module: To display outputs the LCD Advanced Module is used. It has sufficient display size and brightness could also be adjusted. Since this is a prototype the LCD Advanced module is enough to show the required results. The image of the Component is provided in Figure 3 of Appendix C.

Breadboard: A Breadboard is one of the fundamental components for building electronic circuits and it is ideal for prototype projects like the current project. The Breadboard in this project allows connecting the MQ-135 and LCD on it with wires connected from the Arduino to the Breadboard. The image of the component is provided in Figure 4 of Appendix D.

Power Source: A personal computer is used to power the Arduino via a USB cable. The details of the System Configuration of the computer is provided in Appendix E. As Arduino UNO-R3 can be connected to any device that supports USB. The computer was also used for preparing and analysing the dataset to generate results which are analysed to justify the Analytical framework.

2.2 System Setup

The Circuit Diagram and Connections followed are provided in Figure 5 and the complete system is shown in Figure 6 in Appendix F of Appendices. After completing the connections the device was connected to the Computer. To make the Arduino and MQ-135 sensor work the Code was written using an IDE. (Refer to Appendix G for Source Code Written into Arduino) and the result was seen on the LCD. The operation of the System is shown in Figures 7 and Figure 8 in Appendix H.

2.3 Result Generation Methods

To prepare the dataset for analysis the Carbon Dioxide meter was set up and programmed. Then it was connected and the readings from the System were recorded into a CSV file format. A suitable Programming Language is used to carry out the data visualisation and training the Machine Learning Model which are discussed in detail in the following sections.

2.4 Dataset Description

The data were collected for one month starting from 15^{th} November 2019 to 15^{th} December 2019 and the data collection resulted in a total of 92 records. The attributes of the dataset include "Day", "Date", "Time" and "CO2_PPM". The "CO2_PPM" is the density of Carbon Dioxide in air recorded in Parts Per Million at a particular time of day. To keep records of Carbon Dioxide Concentration levels, the readings were taken at three different periods each day. The times were morning, afternoon and evening. For the Machine Learning Algorithm, the "CO2_PPM" will be the target variable. The data were recorded in Dhaka City of Bangladesh from an indoor environment.

2.5 Programming Languages

Two Programming Languages have been used for the project. To program the Arduino the Arduino Language is used. This language is based on C/C++ functions that are called from the code written (Arduino, 2019). For the Data Visualisation and Machine Learning tasks the Python Programming Language is used. Python is selected because it is currently one of the most popular programming languages for scientific computing purposes as it has a wide variety of scientific libraries (Pedregosa et al., 2011). Furthermore, Python is free to use and it has an easy learning curve due to its simple syntax. It has excellent visualisation and Machine Learning Libraries making Python an ideal tool for this research.

2.6 Integrated Development Environments

To program the Arduino connected to MQ-135, the Arduino Integrated Development Environment (IDE) is used. The Arduino IDE, also known as Arduino

Software provides a text editor used to write the code, a console and so on (Arduino, 2015). The Jupyter Notebook is used for the Data Science part of the project. It is a web-based IDE which is interactive and allows the creation and sharing of code. It is well equipped for Data preparation, Visualisation, Machine Learning and so on (Jupyter, 2019). Refer to Appendix I for the Python Code and Appendix J for the List of Libraries used for the Project.

2.7 Machine Learning Model

To make a future prediction about the Concentration Level of CO2 the Support Vector Regression (SVR) model will be used. The model is based on the concepts of Linear Regression function in a feature space of higher dimension. In SVR the input data are mapped by a non-linear function. Research shows that SVR has been applied across many areas such as time series and financial predictions, the approximation of complex engineering analysis and so on (Basak, Pal and Patranabis, 2007). The mathematical details are found in the research paper of Smola and Schölkopf (2004).

3 Technical and Business Case of the System

3.1 Framework

1. ***Costs***: Reducing costs is a primary target for any business. It is because the saved money could be utilised by the company on other aspects to expand the business.
2. ***Ease of Use***: A User-Friendly System is desired. The ease of use is essential when an IoT based system is being built. The programming language used must also be easy to learn and widely used which would ensure that developers are available and the System should also be accessible for non-technical users.
3. ***Hardware***: At the heart of IoT devices there is a component that controls all the operations in the entire system. Selection of components would prove to be vital in development towards a feasible IoT System.
4. ***Software***: Widely used Software is required for programming the IoT device so that it performs accordingly.
5. ***Reliability of Results***: For driving business decisions the generated results need to be accurate.
6. ***Machine Learning***: Big Data is produced by all organisations nowadays. It is also the driving force of prediction services and is now being used in IoT devices. The performance of the prediction task has to be reliable before such a system is deployed on a large scale.
7. ***Expandability***: It is crucial because like Big Data and AI, Cloud Computing is also being used in IoT based Applications. The proposed system should have functionalities to ensure that it could be connected to the Cloud for Data Processing.

3.2 Analysis of the Results

Considering the expenses of building the device it took less then what customers need to pay to purchase such a Carbon Dioxide Monitoring System. The core of the System consists of Arduino UNO-R3. It is an open-source and accessible microcontroller device and is also compatible with a variety of sensors. As described previously the required functionalities are present in the Arduino board. A USB cable is needed to plug the device into the computer to provide power. The device is inexpensive and it is possible to program Arduino with the Arduino IDE which is free software. Programming the Arduino is simple due to large community support resulting in the availability of vast source codes (Kaur et al, 2016). Arduino is not only designed for the technical audience rather for any user as the designer of Arduino aims to provide usability to users from different backgrounds (Diakopoulos and Kapur, 2011). The sensor chosen for the project is the MQ-135 gas sensor which has several important features which are longevity, low cost, good sensitivity to toxic gases. It is also used in industrial gas alarm systems as well as domestic gas alarm system and also provides portability. The LCD Display used is quite sufficient for the project because of its low cost and displaying the required results.

For Data Visualisation and Machine Learning tasks Python is used. Like the Arduino IDE and the Arduino Language, Python is also open source and the Jupyter Notebook is free to use. Among other tools such as R, SAS, MATLAB and so on Python is renowned for its large and active scientific computing community. Adoption of Python for scientific computing for academic research and industries have increased drastically since the early 2000s. Although for Data Analysis and Visualisations it does have strong competition however, Python does have some libraries that make it a suitable Programming Language to perform the Data Exploration and Visualisation tasks (McKinney, 2012). For Machine Learning tasks Python is ideal because of the "scikit-learn" library. This library is currently one of the best Machine Learning Library available (Raschka, 2015).

The statistics of the collected Dataset are shown in Figures 9, 10 and 11. From the Data Analysis, it was found that the maximum CO2 Level recorded was in the 15^{th} of November (Figure 12) whereas the Minimum concentration was recorded on the 17^{th} of November (Figure 13). The overall distribution of Carbon Dioxide Levels is illustrated by a histogram shown in Figure 14. It is found that the overall CO2 Levels were higher in Thursdays as shown in Figure 15 and during the mornings as shown in Figure 16. From Figure 17 it is further observed that the Levels of CO2 had increased slightly then the levels dropped down for some period and increased towards the end of the data collection period.

For predicting the CO2 Levels the Dataset was prepared into a new Dataset (Figure 18). For training the Support Vector Regression Model, the data were divided into training and test sets. The Test Set is shown in Figure 19. After implementing and training SVR on the Training Set, the model was trained on the Test Set and the results are shown in Figure 20. To test the trained model for a future prediction of the CO2 Level of Thursday Morning was generated by

passing in a Dataframe as shown in Figure 21 and the result is shown in Figure 22.

4 Conclusions and Recommendations

In this paper, an IoT based prototype system has been proposed for Sensing and Predicting the Concentration Levels of Carbon Dioxide. The core of this project consists of the Arduino UNO-R3 to generate the readings. The Arduino board is connected to the MQ-135 gas sensor which is used as the sensor to detect CO2 Levels in air. For displaying the readings the Arduino board was connected to an LCD and a personal computer was used to power the Arduino. The system is inexpensive as the designed system is cheaper compared to CO2 meters available. The System was easy to build and code as there are plenty of resources available. From the analysis of the results, it could be inferred that the results are reliable as it is seen that the CO2 Levels had increased a bit which is proved by the fact that the Air Quality Index of Dhaka City was worst in the world in November. In terms of expandability and prediction of the system, there are rooms for future extension of the research.

It is the tasks of IoT devices to handle data. With the generation of data, organisations could make use of such data to generate useful insights and make predictions with the use of AI. The data visualisation results look promising, however, the prediction of the model is weak as seen from the assessment results of SVR in Figure 23. For a future extension of the current research, prolonged data collection is required to identify trends in the data and a stronger Algorithm such as the Neural Network should be used to make better predictions. Cloud Computing is also being adopted by organisations therefor in the future to increase the scalability of the system, the IoT device should be connected to the cloud so that the data could be processed in the Cloud. The Data Analysis, Visualisation and Prediction tasks were done using the Computer System used for the research although such tasks could be done in the Cloud and the results could be shown in different connected devices. Arduino boards are ideal for prototype projects and they could be deployed commercially but Raspberry Pi should be used instead as it is much stronger hardware compared to the Arduino and which would be better for commercial deployment of the device. Furthermore, a larger Display screen should be used and extra data such as Day, Date, Time and predicted PPM of CO2 for the next period of the day should be displayed which would make the device an attractive product and company designing such a product could have a stronger competitive edge over their rivals due to the products innovative functionalities.

References

1. Lehto, J. (2018). *Why Is Measuring CO2 Important?* [online] Vaisala. Available at: `https://www.vaisala.com/en/blog/2019-06/why-measuring-co2-important` [Accessed 8 Dec. 2019].
2. Bonino, S. (2016). *Carbon Dioxide Detection and Indoor Air Quality Control – Occupational Health & Safety.* [online] Occupational Health & Safety. Available at: `https://ohsonline.com/Articles/2016/04/01/Carbon-Dioxide-Detection-and-Indoor-Air-Quality-Control.aspx?Page=1` [Accessed 8 Dec. 2019].
3. Pal, P., Gupta, R., Tiwari, S. and Sharma, A., 2017. IoT based air pollution monitoring system using Arduino. *International Research Journal of Engineering and Technology (IRJET).*
4. Kaur, N., Mahajan, R., Bagai, D. and Student, P.G., 2016. Air quality monitoring system based on Arduino microcontroller. *International Journal of Innovative Research in Science, Engineering and Technology, 5(6).*
5. Arduino.cc. (2019). *Arduino - FAQ.* [online] Available at: `https://www.arduino.cc/en/main/FAQ` [Accessed 18 Dec. 2019].
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research,* 12(Oct):2825–2830.
7. Arduino. (2015). *Arduino - Environment.* [online] Available at: `https://www.arduino.cc/en/guide/environment` [Accessed 19 Dec. 2019].
8. Jupyter. (2019). *Project Jupyter.* [online] Available at: `https://jupyter.org/` [Accessed 19 Dec. 2019].
9. Basak, D., Pal, S. and Patranabis, D.C., 2007. Support vector regression. *Neural Information Processing-Letters and Reviews, 11*(10), pp.203-224.
10. Smola, A.J. and Schölkopf, B., 2004. A tutorial on support vector regression. *Statistics and computing, 14*(3), pp.199-222.
11. Diakopoulos, D. and Kapur, A., 2011, June. HIDUINO: A firmware for building driverless USB-MIDI devices using the Arduino microcontroller. In *NIME* (pp. 405-408).
12. McKinney, W., 2012. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.".
13. Raschka, S., 2015. *Python machine learning.* Packt Publishing Ltd.

Appendices

Appendix A: Arduino UNO-R3

Fig. 1. Arduino board used for the Project

Appendix B: MQ-135

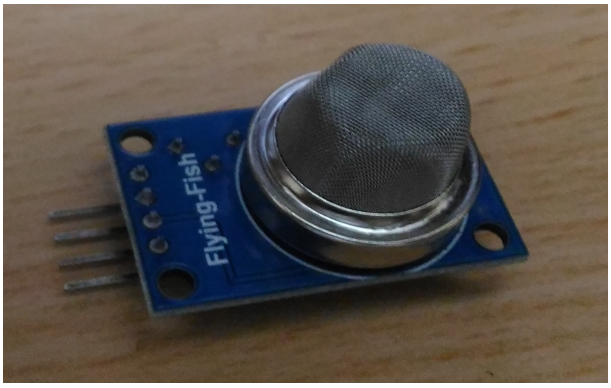


Fig. 2. Gas Sensor used for the Project

Appendix C: LCD Advanced

Fig. 3. LCD used for the Project

Appendix D: Breadboard

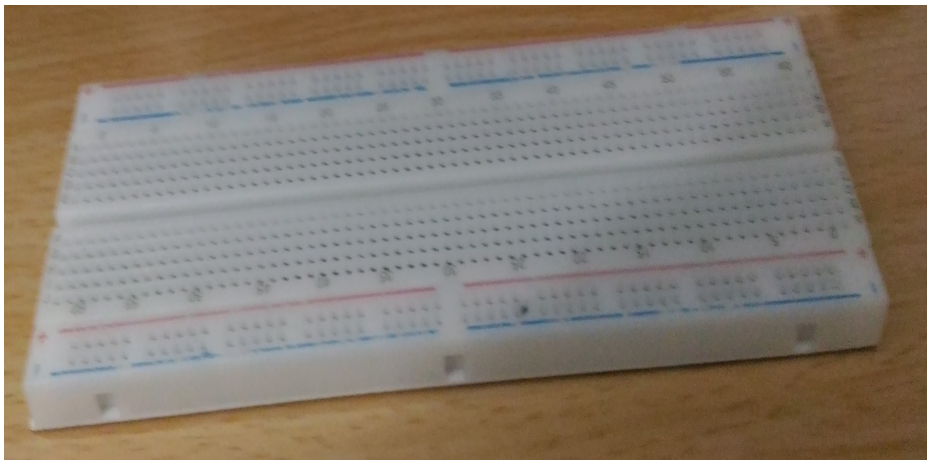


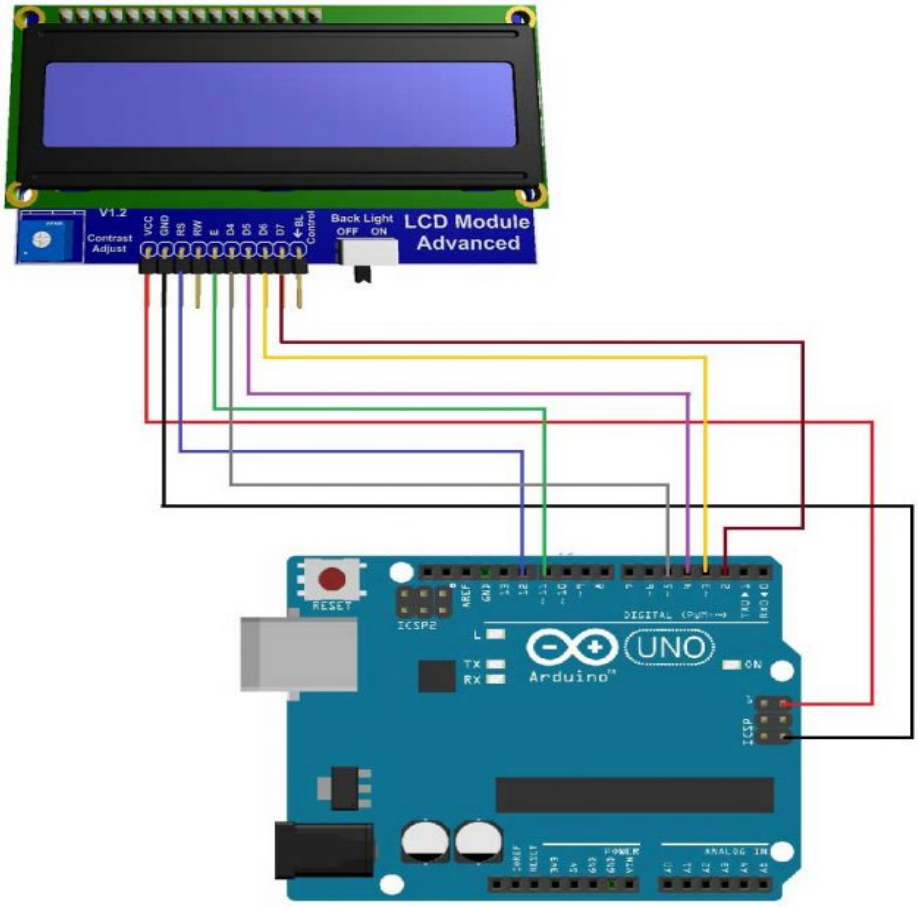
Fig. 4. Breadboard used for the Project

Appendix E: System Configuration

- ***CPU:*** Intel Core i5-9400F
- ***RAM:*** 8GB GDDR4-2400MHz
- ***GPU:*** Nvidia GeForce GTX 1050 Ti
- ***HDD:*** 1TB
- ***SSD:*** 120GB

Appendix F: System Setup

Arduino UNO-R3	MQ-135
5V	VCC
GND	GND
A0	A0



Arduino UNO-R3	LCD module advanced
12	RS
11	En
D4	5
D5	4
D6	3
D7	2
VCC	VDD
GND	VSS

Fig. 5. Circuit Diagram and Connection Details

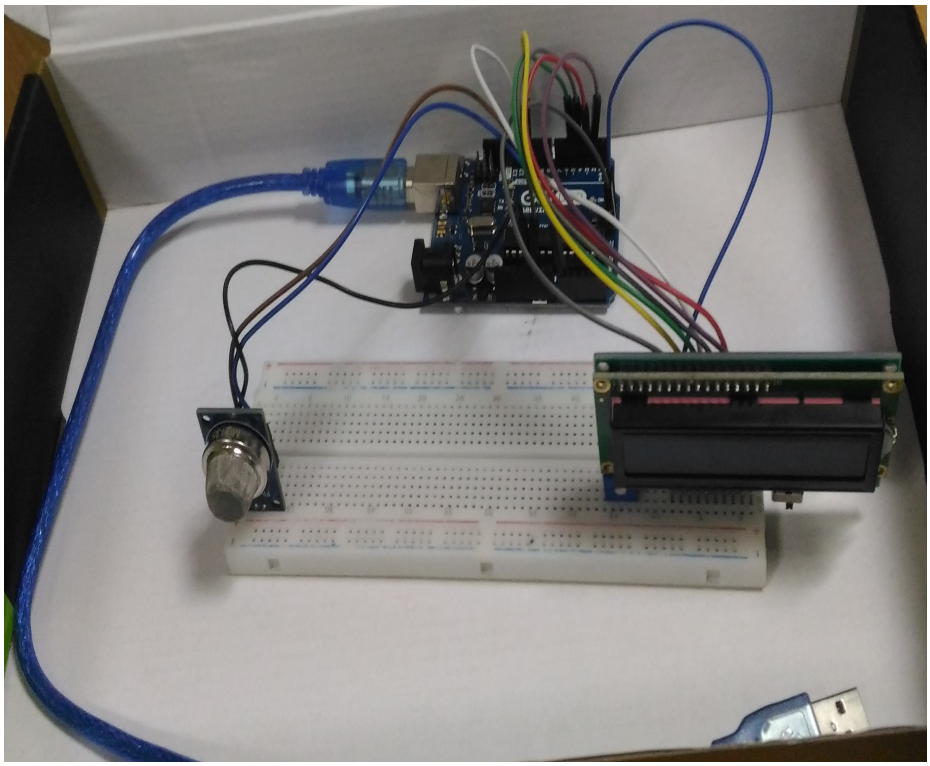


Fig. 6. Complete System

Appendix G: Code Written into Arduino

```
#include <LiquidCrystal.h>

long RL=1000;
long R0=190144;
float a=-.455;
float c=3.02174;
// the setup routine runs once when you press reset:
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7
   = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  // initialize serial communication at 9600 bits per
   second:
  lcd.begin(16, 2);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  long RS=(1024*RL/sensorValue-1000);
  float ratio=(float) RS/(float) R0;
  float alnppm=ratio-c;
  float lnppm=(float)alnppm/a;
  float ppm=exp(lnppm);
  Serial.print(sensorValue);
  Serial.print(",");
  Serial.print("PPM=");
  Serial.println(ppm);
  lcd.setCursor(0, 0);
  // print the number of seconds since reset:
  lcd.print("C02 LEVEL");
  lcd.setCursor(0, 1);
  lcd.print(ppm);
  lcd.print(" PPM");
  delay(800);  // delay in between reads for stability
}
```

Appendix H: Operation of the System

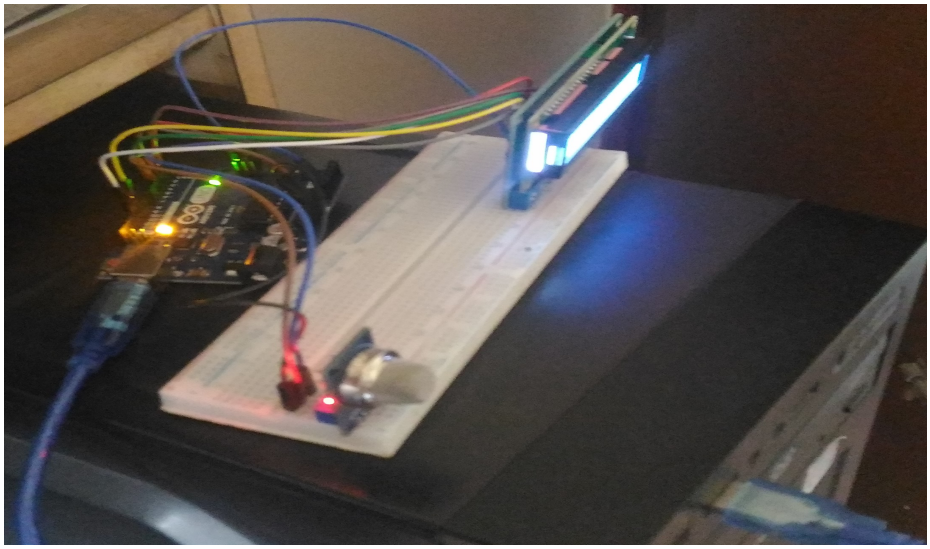


Fig. 7. System Connected to the Computer

Fig. 8. Reading Being Displayed

Appendix I: Python Code

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 16 11:13:52 2019

@author: Dipto
"""
#Loading Important Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Reading the Dataset
df = pd.read_csv("records_backup.csv")

#Statistics
print("Descriptive Statistics About the Dataset")
df.head()
df.shape
df.info()
df.describe(include='all')

#Maximum Concentration Recorded
print("Maximum CO2 PPM Recorded: ","\n",df.loc[df['
    CO2_PPM'].idxmax()])

#Minimum Concentration Recorded
print("Minimum CO2 PPM Recorded: ","\n",df.loc[df['
    CO2_PPM'].idxmin()])

#Distribution of CO2 PPM
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
plt.title(label="Distribution of Carbon Dixoide in
    Parts Per Million",loc="center")
plt.ylabel(ylabel="Frequency")
sns.distplot(df['CO2_PPM'], kde=False,bins=10)

#Distribution of CO2 PPM in Days of the Week
sns.set(style='darkgrid')
plt.figure(figsize=(20,10))
plt.title(label="Concentration of Carbon Dioxide During
     the Days")
sns.barplot(y="Day",x="CO2_PPM",data=df)
```

```
#Changing the Conditions for the "Time" feature for
    Visualisation and Machine Learning
# slice first and second string from time column
df['Hour'] = df['Time'].str[0:2]
df['Hour'] = df.Hour.str.replace(':','')

# convert new column to numeric datetype
df['Hour'] = pd.to_numeric(df['Hour'])

# cast to integer values
df['Hour'] = df['Hour'].astype('int')

# define a function that turns the hours into daytime
    groups
def when_was_it(hour):
    if hour >= 0 and hour < 12:
        return "1"
    elif hour >= 12 and hour < 18:
        return "2"
    elif hour >= 18 and hour < 24:
        return "3"

# create a little dictionary to later look up the
    groups I created
daytime_groups = {1: 'Morning: Between 9:00 and 12:00',
                  2: 'Afternoon: Between 12:01 and
    18:01',
                  3: 'Evening: Between 18:01 and 24:00'
    }

# apply this function to the temporary "Hour" column
df['Daytime'] = df['Hour'].apply(when_was_it)
df['Daytime'] = df['Daytime'].astype('int')


#CO2 PPM during Daytimes
plt.title(label="CO2 Concentrations During Different
    Daytimes")
sns.pointplot(x='Daytime',y='CO2_PPM',data=df)
plt.xlabel(xlabel="1: Morning, 2: Afternoon, 3: Evening
    ")
plt.show()
```

```
#Creating a Decimal Date Column to convert the Date
df['Date']= pd.to_datetime(df['Date'])

#Importing Required Libraries
from datetime import datetime as dt
import time

#Function for Convertion to Decimal Date
def toYearFraction(date):
    def sinceEpoch(date): # returns seconds since epoch
        return time.mktime(date.timetuple())
    s = sinceEpoch

    year = date.year
    startOfThisYear = dt(year=year, month=1, day=1)
    startOfNextYear = dt(year=year+1, month=1, day=1)

    yearElapsed = s(date) - s(startOfThisYear)
    yearDuration = s(startOfNextYear) - s(
   startOfThisYear)
    fraction = yearElapsed/yearDuration

    return date.year + fraction


#Creation of Decimal_Date column
df["Decimal_Date"] = df['Date'].apply(toYearFraction)

#Levels of CO2 During the Month
plt.title(label="Levels of CO2 During the Month")
sns.lineplot(x="Decimal_Date",y="CO2_PPM",data=df)
plt.show()

#Creating a Dataframe for Machine Learning
ml_df = df.drop(columns=["Date","Hour","Time"])

#Dummy Encoding
ml_df = pd.get_dummies(data=ml_df,drop_first=True)

# Defining the features
x = ml_df.drop(['CO2_PPM'], axis=1)

# Defining the target Variable
y = ml_df[['CO2_PPM']]
```

```
#Splitting the Training and Test Set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y
    ,test_size=0.2,random_state=42)


#Making Predictions
from sklearn import svm
clf = svm.SVR(kernel="rbf",gamma="scale")
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

#Evaluating Model Performance
from sklearn import metrics
print('Mean Absolute Error:', metrics.
    mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error
    (y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.
    mean_squared_error(y_test, y_pred)))

data={'Daytime':[1],'Decimal_Date':[2019.873973],'
    Day_Monday':[0],'Day_Saturday':[0],'Day_Sunday'
    :[0],
      'Day_Thursday':[1],'Day_Tuesday':[0],'
    Day_Wednesday':[0],}
pred = pd.DataFrame(data)
print(pred)

print("Prediction of CO2 PPM for Tomorrow Morning:")
clf.predict(pred)
```

Appendix J: Libraries Used

LiquidCrystal: This library allows an Arduino/Genuino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4 or 8 bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

NumPy: Short for Numerical Python, it is the foundation package used for high performance computing and data analysis. Numpy is used for Linear Algebra as the matrix of features (X), the dependent variable (y) and matrices for training and testing samples of data are stored using the array provided by NumPy.

Pandas: The pandas library contains high-level data structures built on top of NumPy that makes data analysis fast and easy in Python. For this research the entire data set will be stored in a data structure provided by the Pandas library.

Matplotlib: The "pyplot" class of this library will be used to plot figures where required.

Seaborn: It is A library build on top of Matplotlib and it allows drawing attractive and informative statistical graphics.

Scikit-learn: It provides the state of the art implementation of all the well known Machine Learning algorithms provided with easy to use interface integrated with Python. This library also has built in classes and methods that are used for performance evaluation of Machine Learning models.

datetime and time: The modules are used for manipulating dates and times in the Dataset. The "Date" feature has been converted to "Decimal_Date" using the Modules.

Appendix K: Results of Data Visualisation and Prediction

Descriptive Statistics About the Dataset

	Day	Date	Time	CO2_PPM
0	Friday	15/11/2019	11:53:00	726.81
1	Friday	15/11/2019	19:00:00	746.14
2	Saturday	16/11/2019	11:21:00	707.12
3	Saturday	16/11/2019	12:07:00	692.20
4	Saturday	16/11/2019	18:11:00	744.09

Fig. 9. First Five Data Records

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 4 columns):
Day         92 non-null object
Date        92 non-null object
Time        92 non-null object
CO2_PPM     92 non-null float64
dtypes: float64(1), object(3)
memory usage: 3.0+ KB
```

Fig. 10. Dataset Description

	Day	Date	Time	CO2_PPM
count	92	92	92	92.000000
unique	7	31	83	NaN
top	Sunday	26/11/2019	19:00:00	NaN
freq	15	3	2	NaN
mean	NaN	NaN	NaN	722.840761
std	NaN	NaN	NaN	20.760240
min	NaN	NaN	NaN	614.420000
25%	NaN	NaN	NaN	717.852500
50%	NaN	NaN	NaN	729.045000
75%	NaN	NaN	NaN	734.680000
max	NaN	NaN	NaN	746.140000

Fig. 11. General Statistics of the Features

```
Maximum CO2 PPM Recorded:
 Day                 Friday
Date          15/11/2019
Time            19:00:00
CO2_PPM           746.14
Name: 1, dtype: object
```

Fig. 12. Maximum Concentration Recorded

```
Minimum CO2 PPM Recorded:
  Day                Sunday
Date           17/11/2019
Time             19:08:00
CO2_PPM            614.42
Name: 7, dtype: object
```

Fig. 13. Minimum Concentration Recorded

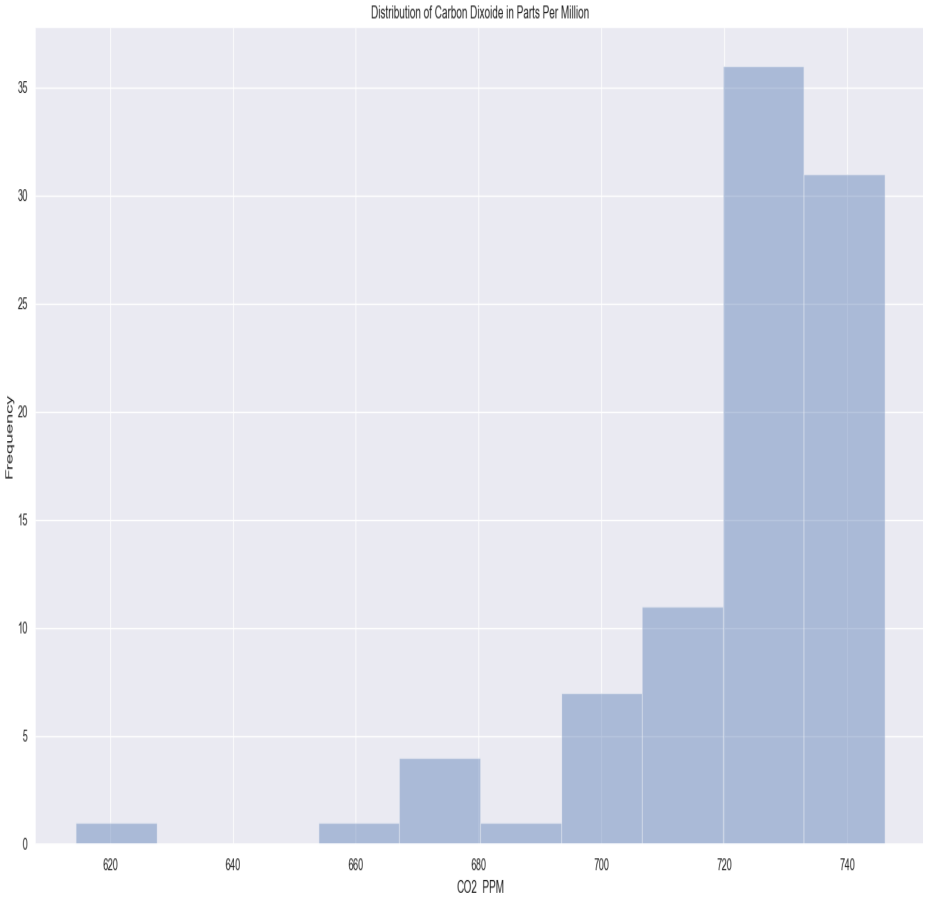


Fig. 14. Distribution of CO2 Concentration

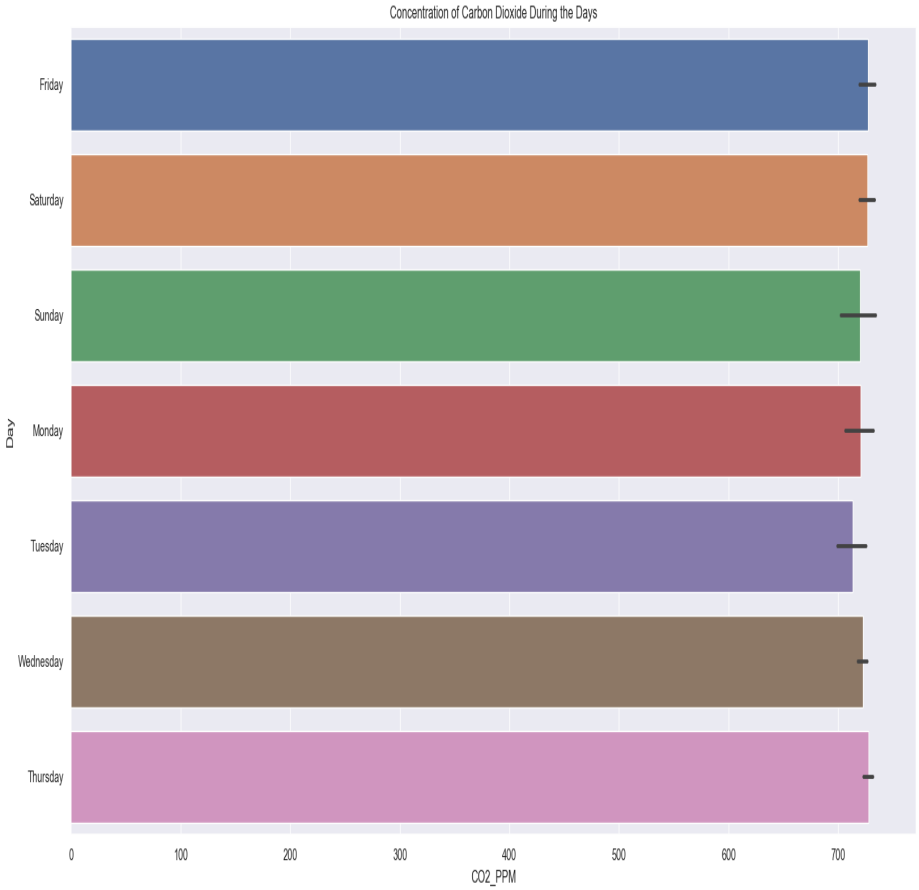


Fig. 15. Concentration of CO2 During the Days

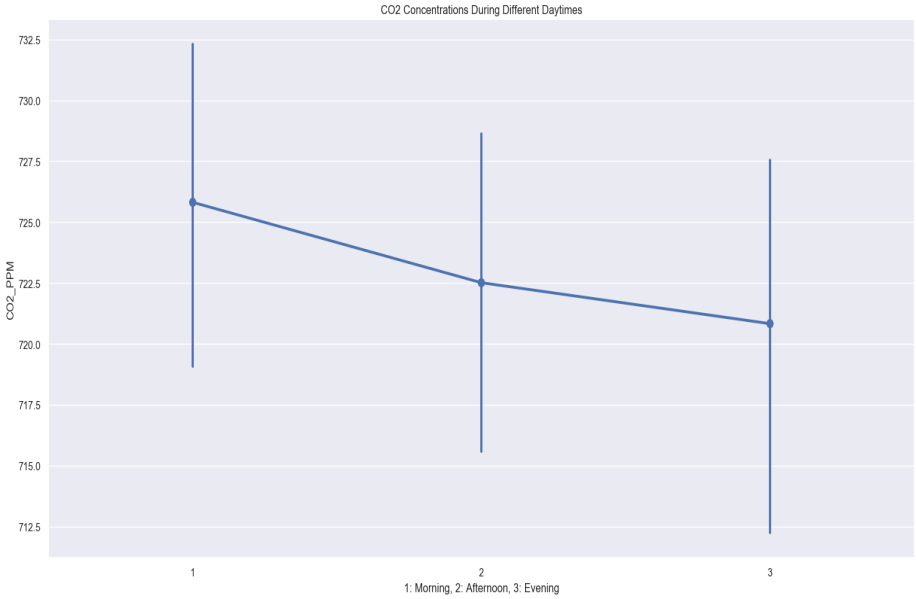


Fig. 16. Concentration of CO2 During Different Times of the Days

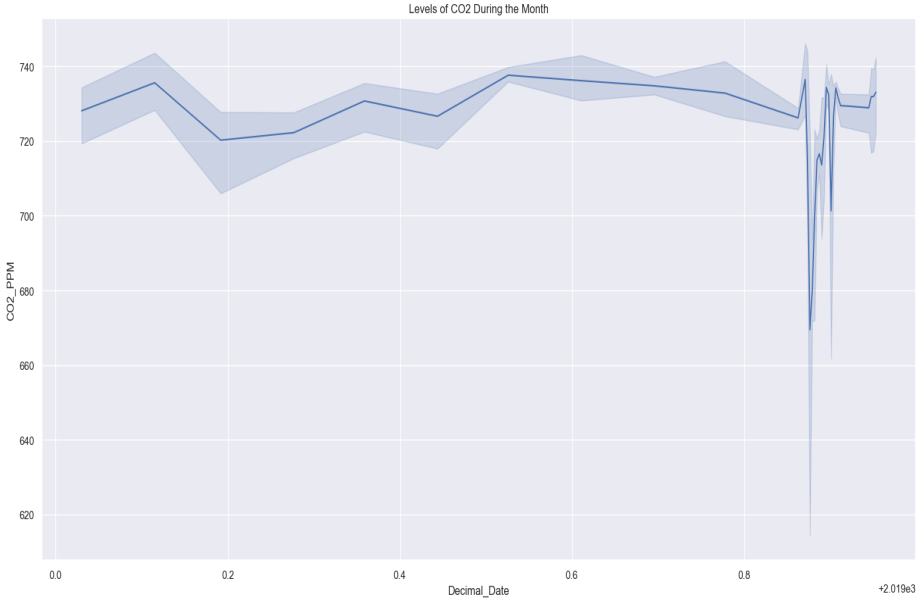


Fig. 17. Concentration of CO2 with Date

	CO2_PPM	Daytime	Decimal_Date	Day_Monday	Day_Saturday	Day_Sunday	Day_Thursday	Day_Tuesday	Day_Wednesday
0	726.81	1	2019.871233	0	0	0	0	0	0
1	746.14	3	2019.871233	0	0	0	0	0	0
2	707.12	1	2019.873973	0	1	0	0	0	0
3	692.20	2	2019.873973	0	1	0	0	0	0
4	744.09	3	2019.873973	0	1	0	0	0	0

Fig. 18. Dataset Prepared for SVR

	CO2_PPM
40	735.03
22	731.65
55	728.58
72	732.44
0	726.81
26	732.44
39	731.65
67	735.88
10	693.76
44	731.85
83	739.49
35	734.68
89	742.27
62	732.64
12	723.09
4	744.09
18	715.74
28	729.95
49	730.60

Fig. 19. The Test Set

```
array([727.52999914, 727.5299986 , 727.52999738, 727.53000617,
       727.53000648, 727.530007  , 727.53000306, 727.52999969,
       727.53000198, 727.53000698, 727.53000637, 727.53000363,
       727.53000692, 727.5300071 , 727.5300003 , 727.52999919,
       727.53000309, 727.52999915, 727.53000042])
```

Fig. 20. Predictions Made by SVR on the Test Set

	Daytime	Decimal_Date	Day_Monday	Day_Saturday	Day_Sunday	Day_Thursday	Day_Tuesday	Day_Wednesday
0	1	2019.873973	0	0	0	1	0	0

Fig. 21. Dataset Passed into the Trained Model

```
Prediction of CO2 PPM for Tomorrow Morning:

array([727.53000703])
```

Fig. 22. Prediction Made by SVR

```
Mean Absolute Error: 7.947893213151647
Mean Squared Error: 118.66366771817785
Root Mean Squared Error: 10.893285441875552
```

Fig. 23. Assessment of SVR