Air Quality Monitoring

**Project Proposal: IoT Air Quality Monitoring System**

# **Project Overview:**

AIR QUALITY MONITORING SYSTEM. Protecting the atmospheric environment involves control of atmospheric emissions as well as an understanding of pollutant dispersion, monitoring emission levels, i.e. concentration in ambient air. To monitor these levels there are Air Quality Monitoring Networks.

## **Objectives**

The main objectives stated for the development of an air quality measurement and surveillance programme might be to: − facilitate the background concentration(s) measurements, − monitor current levels as a baseline for assessment, − check the air quality relative to standards or limit values, − detect the importance of individual sources, − enable comparison of the air quality data from different areas and countries, − collect data for the air quality management, traffic and land-use planning purposes, − observe trends (related to emissions), − develop abatement strategies, − determine the exposure and assess the effects of air pollution on health, vegetation or building materials, − inform the public about the air quality and raise the awareness, − develop warning systems for the prevention of undesired air pollution episodes, − facilitate the source apportionment and identification, − supply data for research investigations, − develop/validate management tools (such as models), − develop and test analytical instruments and − to support legislation in relation to the air quality limit values and guidelines.

## Scope

* Deployment of IoT sensors in selected public areas.
* Development of a Python script for data collection and transmission.
* Integration with a data-sharing platform.
* User-friendly data visualization for water consumption monitoring.
* Testing and validation of the system's accuracy and reliability

# **Hardware**

**Microcontrollers:**

They are used to interface with the IoT sensors, process data, and facilitate communication with the data-sharing platform.

**Used:** Arduino,Esp32

**Connectivity:**

**Wi-Fi Module**: Use Wi-Fi for data transmission.

# **Sensors**

We have used the following sensors for our project:

Pid sensor

CO2 sensor

## **Their uses**

PID sensor :

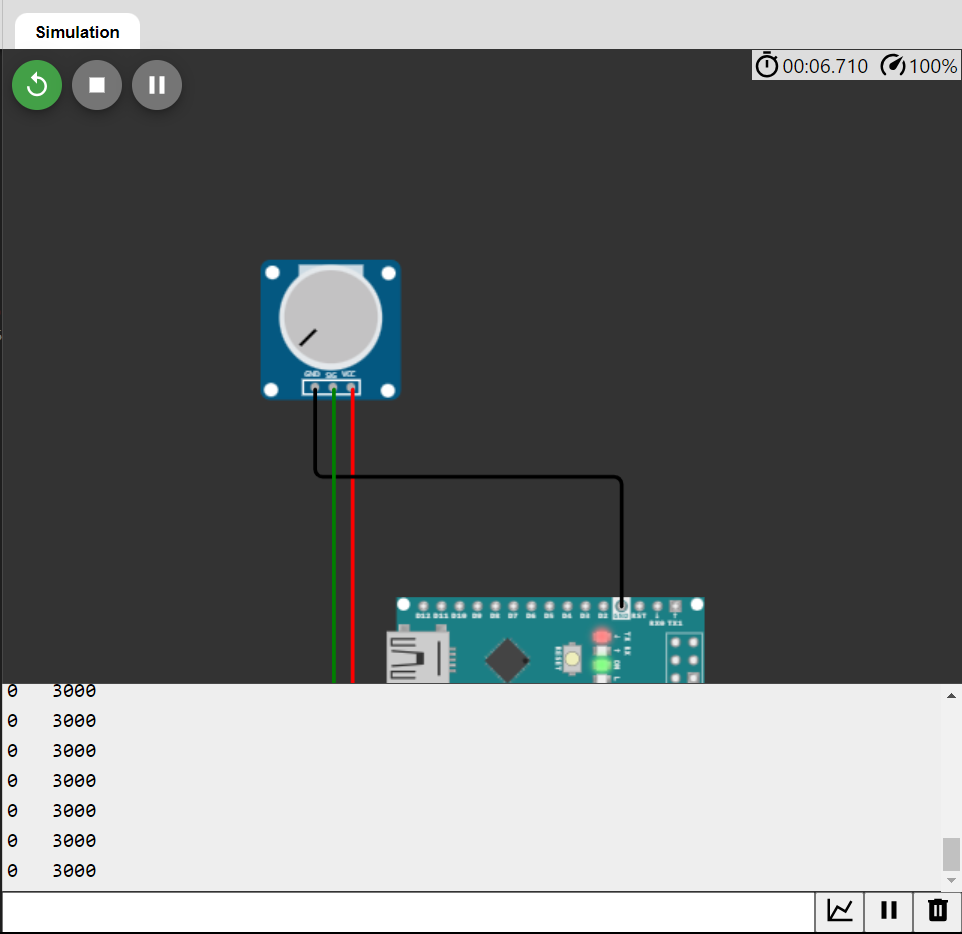
A Photoionization Detector (PID) is a gas detector used to measures volatile organic compounds (VOCs), such as benzene, and other gases.

CO2 sensor:

CO2 sensors are used to monitor fermentation, respiration, photosynthesis, and other carbon dioxide consuming or producing processes. They can be used for HVAC applications to monitor air quality. There are wider applications for CO2 sensors in the agricultural, food, pharmaceutical, refrigeration and brewing sectors.

Screenshots:

PID sensor



# Script for data sharing

Python script for sharing data from PID sensor

|  |
| --- |
| import time |
| import os.path  channel\_id = " 2311196"  write\_api\_key = "BUDE9LL0TGY28R1D "  thing\_speak\_url = "https://api.thingspeak.com/channels.json?api\_key=BUDE9LL0TGY28R1D" |
|  |
| from OmegaExpansion import AdcExp |
| from OmegaExpansion import pwmExp |
|  |
| pwmExp.setVerbosity(-1) |
| pwmExp.driverInit() |
| adc = AdcExp.AdcExp() |
|  |
| targetT = 35 |
| P = 10 |
| I = 1 |
| D = 1 |
|  |
| pid = PID.PID(P, I, D) |
| pid.SetPoint = targetT |
| pid.setSampleTime(1) |
|  |
| def readConfig (): |
| global targetT |
| with open ('/tmp/pid.conf', 'r') as f: |
| config = f.readline().split(',') |
| pid.SetPoint = float(config[0]) |
| targetT = pid.SetPoint |
| pid.setKp (float(config[1])) |
| pid.setKi (float(config[2])) |
| pid.setKd (float(config[3])) |
|  |
| def createConfig (): |
| if not os.path.isfile('/tmp/pid.conf'): |
| with open ('/tmp/pid.conf', 'w') as f: |
| f.write('%s,%s,%s,%s'%(targetT,P,I,D)) |
|  |
| createConfig() |
|  |
| while 1: |
| readConfig() |
| #read temperature data |
| a0 = adc.read\_voltage(0) |
| temperature = (a0 - 0.5) \* 100 |
|  |
| pid.update(temperature) |
| targetPwm = pid.output |
| targetPwm = max(min( int(targetPwm), 100 ),0) |
|  |
| print "Target: %.1f C | Current: %.1f C | PWM: %s %%"%(targetT, temperature, targetPwm) |
|  |
| # Set PWM expansion channel 0 to the target setting |
| pwmExp.setupDriver(0, targetPwm, 0)  TIMT.SLEEP(0.5)  while True:      try:            pid\_sensor\_rate = pid\_sensor\_data()            data = {              "field1": pid\_sensor\_rate          }          response = requests.post(thing\_speak\_url, data=data)          print("Data sent to ThingSpeak. Status code:", response.status\_code)      except Exception as e:          print("Error:", str(e))      time.sleep(15) |

Python script for sharing data fromCO2 Sensor

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

import co2meter

channel\_id = " 2311196"

write\_api\_key = "BUDE9LL0TGY28R1D "

thing\_speak\_url = "https://api.thingspeak.com/channels.json?api\_key=BUDE9LL0TGY28R1D"

while True:

tb = time.time()

mon = co2meter.CO2monitor()

now = pandas.Timestamp.now()

output\_filename = "{}-{}-{}.csv".format(now.year, now.month, now.day)

if not os.path.exists(output\_filename):

with open(output\_filename, "w") as f:

f.write("Time,Concentration,Temperature\n")

data = mon.read\_data()

t = time.mktime(data.index[0].timetuple())

row = t, np.float64(data["co2"]), np.float64(data["temp"])

print("{}, {} PPM, {} °C".format(\*row))

with open(output\_filename, "a") as f:

writer = csv.writer(f)

writer.writerow(row)

make\_plot()

tsleep = 60 - (time.time() - tb)

if tsleep > 0:

time.sleep(tsleep)

while True:

    try:

        CO2\_sensor\_rate = CO2\_sensor\_data()

data = {

            "field2": CO2\_sensor\_rate

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)