Date	18 November 2022
Team ID	PNT2022TMID48655
Project Name	Gas Leakage monitoring and alerting system for industries

SOURCE CODE OF THE DEVELOPED SYSTEM USING PYTHON:

```
#!/usr/bin/env python3

import RPi.GPIO as GPIO import time
import math

import Adafruit_ADS1x15

adc = Adafruit_ADS1x15.ADS1115()
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
buzzer=7 led_red=22
from twilio.rest import Client

account_sid = 'AC166345fc028fb1bc9208c50e11e339d1'
auth_token = '31b69bac1e5a37e304246beeab8f2bfb'
client = Client(account_sid, auth_token)
GPIO.setup(buzzer,GPIO.OUT)
GPIO.output(buzzer,0)
GPIO.setup(led_red,GPIO.OUT)
 GPIO.output(led_red,0) led_green=37
c=0
d=0
e=0
f=0
GPIO.setup(led_green,GPIO.OUT)
GPIO.output(led_green,GPIO.LOW)

import socket
from select import select def recv(sock):
data = "" while True:
cData = rs.recv(1024).decode()
if cData == "\r\n":
            break

data += cData.rstrip().casefold()

return data
```

```
def sensor_data():
pin_0=adc.read_adc(0, gain=GAIN) pin_1=adc.read_adc(1, gain=GAIN)
pin_2=adc.read_adc(2, gain=GAIN) pin_3=adc.read_adc(3, gain=GAIN)
pin_0=pin_0*0.02048 pin_1=pin_1*0.02048 pin_2=pin_2*0.02048
            pin_3=pin_3*0.02048

return (pin_0,pin_1,pin_2,pin_3)
def buzzer_ring():

GPIO.output(buzzer,1) time.sleep(0.8) GPIO.output(buzzer,0) time.sleep(0.4)

print("Buzzer is in working form")
def led_testing():
GPIO.output(led_red,1) time.sleep(0.8) GPIO.output(led_red,0) time.sleep(0.4)
def message():

message = client.messages.create(body="Gas has Been
Detected",from_='+14437753694',to='+905338554396')
def message_for_all_set():

message = client.messages.create(body="Everything is under
control",from_='+14437753694',to='905338554396')
print("[+] Initializing Server Socket")

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
host = "0.0.0.0"

port = 6000

sock.bind((host, port)) sock.listen(5)
print("[+] Server listening on {}:{}".format(host, port))

GAIN = 2

connection = [sock]

while True:

readSock ,writeSock, errorSock = select(connection, connection, [])

for rs in readSock:
if rs == sock:

clientSock, clientAddr = sock.accept() connection.append(clientSock)
print("[+] Client Connected: {}".format(clientAddr))

else:
```

```
try:
data=rs.recv(1024).decode().rstrip().casefold()
 print("[{}]: {}".format(rs.getpeername()[0], data))
if rs in writeSock:
if "login" in data:
while True:
(pin_0,pin_1,pin_2,pin_3)=sensor_data()
print("This is output from Sensor 1 ",pin_0)
print("This is output from the sensor 2 ",pin_1)
print("This is output from the Sensor 3 ",pin_2)
print("This is output from the sensor 4 ",pin_3)
if pin_0 > 200:
pin_0=(str(pin_0)+"\n")
rs.send("Sensor 1 detected the gas\n".encode()) rs.send(pin_0.encode())
buzzer_ring() led_testing()
if c==0:
message() c=c+1
rs.send("This is the value of Gas Concentration\n".encode())
print("Gas has been detected:\n")
f=0
elif pin_1 > 150: pin_1=(str(pin_1)+"\n")
rs.send("Sensor 2 detected the gas\n".encode()) rs.send(pin_1.encode())
buzzer_ring() led_testing()
if d==0:
message() d=d+1
```

```
rs.send("This is the value of Gas Concentration\n".encode())

f=0

print("Gas has been detected:\n") elif pin_2 > 150:

pin_2=(str(pin_2)+"\n")

rs.send("Sensor 3 detected the gas\n".encode()) buzzer_ring()

led_testing()

if e==0:
message()
e=e+1
rs.send(pin_2.encode())

rs.send("This is the value of Gas Concentration\n".encode())

print("Gas has been detected:\n")

f=0

elif

pin_3 > 150: pin_3=(str(pin_3)+"\n")

rs.send("Sensor 4 detected the gas\n".encode()) rs.send(pin_3.encode())

rs.send("This is the value of Gas Concentration\n".encode())

print("Gas has been detected:\n")

f=0

else:
print("All set\n")

rs.send("All Set\n".encode())

if f==0:

message_for_all_set() f=f+1

GPIO.output(buzzer,GPIO.LOW) GPIO.output(led_red,GPIO.LOW)

GPIO.output(led_green,GPIO.HIGH) time.sleep(0.8)
```

GPIO.output(led_green,GPIO.LOW) time.sleep(0.4)

c=0

d=0

e=0

time.sleep(0.5)

else:

print("Have you been gone mad or drank\n") rs.send("Have you been gone mad or drank\n".encode())
except:

print("[!] Client went offline") GPIO.output(buzzer,GPIO.LOW)

GPIO.output(led_red,GPIO.LOW)

connection.remove(rs)

rs.close()