SPRINT – 1 DELIVERY

Date	November 9, 2022
Team ID	PNT2022TMID26105
Project Name	Real-Time River Water Quality Monitoring and Control System

PYTHON PROGRAM:-

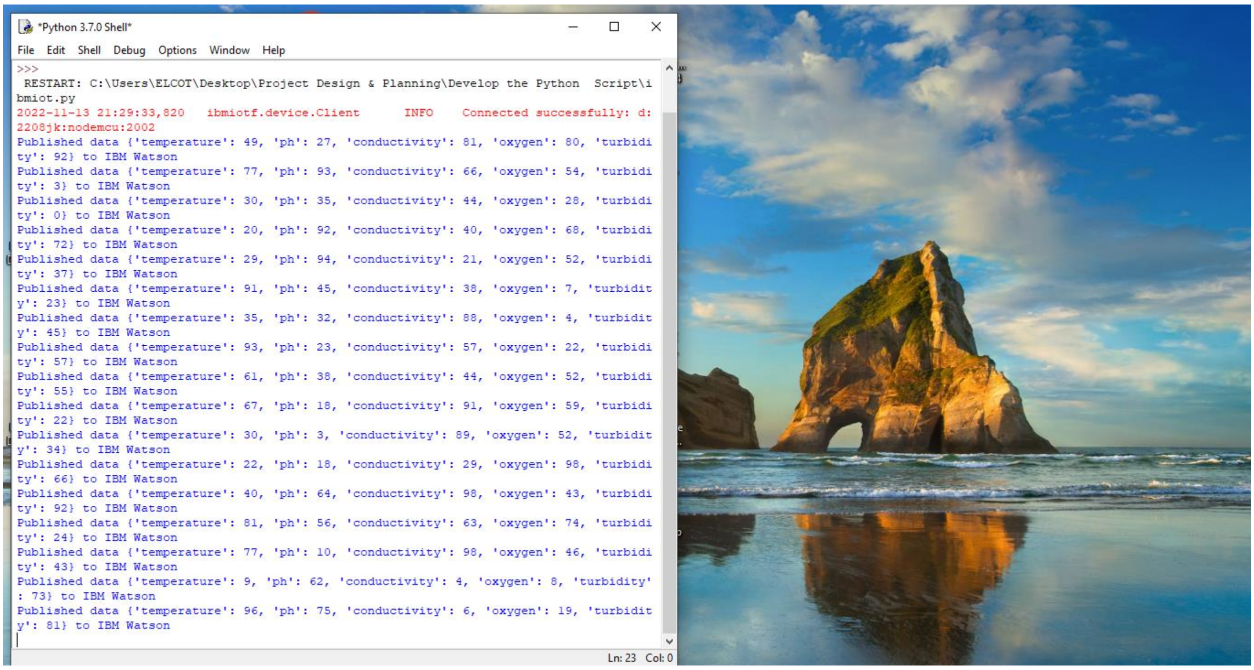
```
import random
import time
import sys
import ibmiotf.application
import ibmiotf.device
```

# Provide your IBM Watson Device Credentials

```
organization = "dymr4l"  # repalce it with organization ID
deviceType = "NodeMCU"  # replace it with device type
deviceId = "2002"  # repalce with device id
authMethod = "token"
authToken = "Nirmal@2002"  # repalce with token


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    if cmd.data['command'] == 'lighton':
        print("LIGHT ON")
    elif cmd.data['command'] == 'lightoff':
        print("LIGHT OFF")


try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
authMethod,
                     "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
# ..............................................

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()
```

```
while True:
    pH = random.randint(0,100)
    conductivity = random.randint(0,100)
    T = random.randint(0,100)
    oxygen = random.randint(0,100)
    turbidity = random.randint(0,100)

    # Send Temperature & Humidity to IBM Watson
    data = {'temperature': T,'ph':pH,'conductivity':conductivity,'oxygen':oxygen,"turbidity":turbidity}


    # print data
    def myOnPublishCallback():
        print("Published data",data, "to IBM Watson")


    success = deviceCli.publishEvent("event", "json", data, 0, myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(5)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
```

OUTPUT:



```
>>>
 RESTART: C:\Users\ELCOT\Desktop\Project Design & Planning\Develop the Python  Script\i
bmiot.py
2022-11-13 21:29:33,820   ibmiotf.device.Client      INFO    Connected successfully: d:
2208jk:nodemcu:2002
Published data {'temperature': 49, 'ph': 27, 'conductivity': 81, 'oxygen': 80, 'turbidi
ty': 92} to IBM Watson
Published data {'temperature': 77, 'ph': 93, 'conductivity': 66, 'oxygen': 54, 'turbidi
ty': 3} to IBM Watson
Published data {'temperature': 30, 'ph': 35, 'conductivity': 44, 'oxygen': 28, 'turbidi
ty': 0} to IBM Watson
Published data {'temperature': 20, 'ph': 92, 'conductivity': 40, 'oxygen': 68, 'turbidi
ty': 72} to IBM Watson
Published data {'temperature': 29, 'ph': 94, 'conductivity': 21, 'oxygen': 52, 'turbidi
ty': 37} to IBM Watson
Published data {'temperature': 91, 'ph': 45, 'conductivity': 38, 'oxygen': 7, 'turbidit
y': 23} to IBM Watson
Published data {'temperature': 35, 'ph': 32, 'conductivity': 88, 'oxygen': 4, 'turbidit
y': 45} to IBM Watson
Published data {'temperature': 93, 'ph': 23, 'conductivity': 57, 'oxygen': 22, 'turbidi
ty': 57} to IBM Watson
Published data {'temperature': 61, 'ph': 38, 'conductivity': 44, 'oxygen': 52, 'turbidi
ty': 55} to IBM Watson
Published data {'temperature': 67, 'ph': 18, 'conductivity': 91, 'oxygen': 59, 'turbidi
ty': 22} to IBM Watson
Published data {'temperature': 30, 'ph': 3, 'conductivity': 89, 'oxygen': 52, 'turbidit
y': 34} to IBM Watson
Published data {'temperature': 22, 'ph': 18, 'conductivity': 29, 'oxygen': 98, 'turbidi
ty': 66} to IBM Watson
Published data {'temperature': 40, 'ph': 64, 'conductivity': 98, 'oxygen': 43, 'turbidi
ty': 92} to IBM Watson
Published data {'temperature': 81, 'ph': 56, 'conductivity': 63, 'oxygen': 74, 'turbidi
ty': 24} to IBM Watson
Published data {'temperature': 77, 'ph': 10, 'conductivity': 98, 'oxygen': 46, 'turbidi
ty': 43} to IBM Watson
Published data {'temperature': 9, 'ph': 62, 'conductivity': 4, 'oxygen': 8, 'turbidity'
: 73} to IBM Watson
Published data {'temperature': 96, 'ph': 75, 'conductivity': 6, 'oxygen': 19, 'turbidit
y': 81} to IBM Watson
```

CODE FOR ARDUINO:

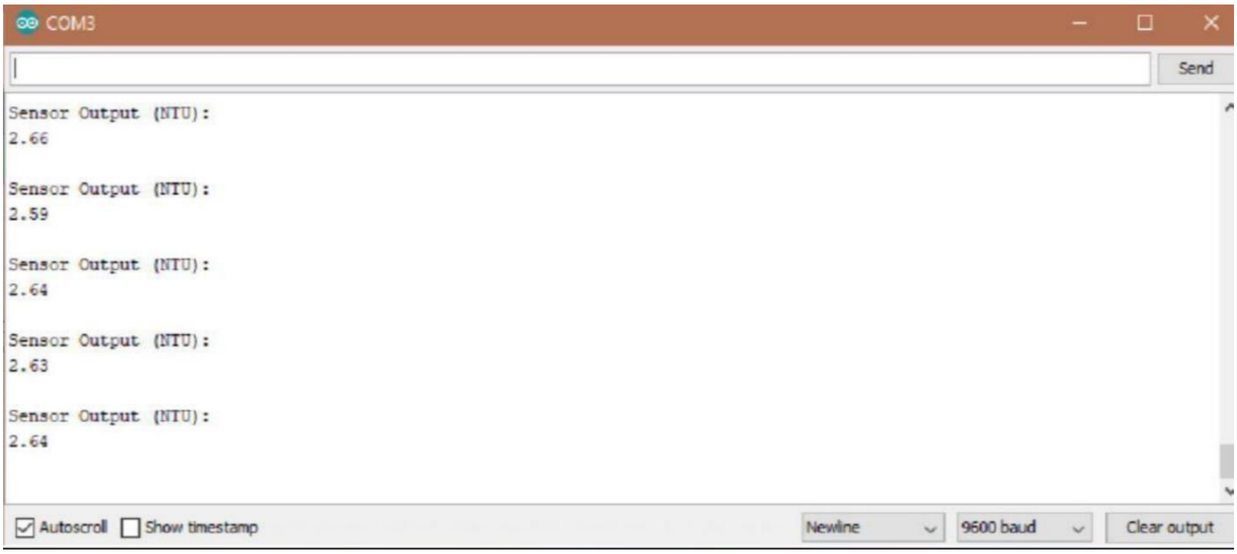
```
#include <OneWire.h>
#include <DallasTemperature.h> #define ONE_WIRE_BUS 5
OneWire oneWire(ONE_WIRE_BUS); DallasTemperature
sensors(&oneWire); float Celcius=0; float Fahrenheit=0; float
voltage=0; const int analogInPin = A0; int sensorValue = 0;
unsigned long int avgValue; float b; int buf[10],temp; void
setup(void)
{

 Serial.begin(9600); sensors.begin(); int sensorValue = analogRead(A1);
voltage = sensorValue * (5.0 /
1024.0);
} void loop(void) { sensors.requestTemperatures();
Celcius=sensors.getTempCByIndex(0);
                                        Fahrenheit=sensors.toFahrenheit(C
elcius); for(int i=0;i<10;i++) { buf[i]=analogRead(analogInPin);
delay(10); } for(int i=0;i<9;i++) { for(int j=i+1;j<10;j++)
{ if(buf[i]>buf[j]) { temp=buf[i]; buf[i]=buf[j];
}
 } } for(int i=2;i<8;i++) avgValue+=buf[i]; float
pHVol=(float)avgValue*5.0/1024/6; float phValue = -5.70 * pHVol +
21.34;

Serial.println(phValue);
Serial.print("pH");



 Serial.print(" C ");
 Serial.print(Celcius);

 Serial.print(voltage); Serial.print("V"); delay(10000);
}
```

CODE IMPLEMENTATION:

```
import serial import time import csv import numpy as np import matplotlib.pyplot as plt ser =
serial.Serial('/COM6',9600) ser_bytes = ser.readline(10) print (ser_bytes) ser.flushInput() while
True:
 try:
 ser_bytes = ser.readline() decoded_bytes = float(ser_bytes[0:len(ser_bytes)-
2].decode("utf-8")) print(decoded_bytes)

temp = float(decoded_bytes(1:3)) turb = float(decoded_bytes(4:6))
pH = float(decoded_bytes(6:8)) with open("test_data.csv","a") as f:
writer = csv.writer(f,delimiter=",")
writer.writerow([time.time(),decoded_bytes]) except:
 {  if(buf[i]>buf[j])  {  temp=buf[i];  buf[i]=buf[j];
print("Keyboard Interrupt") ser.close() break() t = np.arange(0.0, 2.0,

0.01)  s  =  1  +  np.sin(2*np.pi*t)  plt.plot(t,  s)  plt.xlabel('time  (s)')

plt.ylabel('Celsisus  (C)')
plt.title('Temperature') plt.grid(True)  plt.savefig("Temperature.png")
Serial.begin(9600); sensors.begin(); int sensorValue = analogRead(A1);
voltage =sensorValue * (5.0 / 1024.0);
}
void loop(void)
{
 sensors.requestTemperatures();
 Celcius=sensors.getTempCByIndex(0);
                                                  Fahrenheit=sensors.toFahrenheit(C
elcius); for(int i=0;i<10;i++)
{
 buf[i]=analogRead(analogInPin); delay(10);
}
for(int i=0;i<9;i++)
{
 for(int j=i+1;j<10;j++)
 {
 if(buf[i]>buf[j])
 {
 temp=buf[i]; buf[i]=buf[j]; buf[j]=temp;
 }
n = 256
X = np.linspace(-np.pi, np.pi, 256, endpoint=True) C,S = np.cos(X),
np.sin(X) plt.plot(X, C) plt.plot(X,S) plt.show()

print ("Visualization of real time sensor Data.") print("/n") while True:
 try:
 ser_bytes    =    ser.readline()    decoded_bytes    =    float(ser_bytes[0:len(ser_bytes)-
2].decode("utf-8"))   print(decoded_bytes)   temp   =   float(decoded_bytes(1:3))   turb   =
float(decoded_bytes(4:6)) pH = float(decoded_bytes(6:8)) with open("test_data.csv","a")
as f: writer = csv.writer(f,delimiter=",")
```

```
np.arange(0.0, 2.0, 0.01) s = 1 + np.sin(2*np.pi*t) plt.plot(t, s)
plt.title('Temperature') plt.grid(True)  plt.savefig("Temperature.png")
```

ARDUINO OUTPUT:



```
Sensor Output (NTU):
2.66

Sensor Output (NTU):
2.59

Sensor Output (NTU):
2.64

Sensor Output (NTU):
2.63

Sensor Output (NTU):
2.64
```

```
plt.title('Temperature') plt.grid(True)  plt.savefig("Temperature.png")
```

CODE FOR ARDUINO:

```
#include <OneWire.h>
#include <DallasTemperature.h> #define ONE_WIRE_BUS 5
OneWire oneWire(ONE_WIRE_BUS); DallasTemperature
sensors(&oneWire); float Celcius=0; float Fahrenheit=0; float
voltage=0; const int analogInPin = A0; int sensorValue = 0;
unsigned long int avgValue; float b; int buf[10],temp; void
setup(void)
{

Serial.begin(9600); sensors.begin(); int sensorValue = analogRead(A1);
voltage =sensorValue * (5.0 /
1024.0);
} void loop(void) { sensors.requestTemperatures();
Celcius=sensors.getTempCByIndex(0);
                                              Fahrenheit=sensors.toFahrenheit(C
elcius); for(int i=0;i<10;i++) { buf[i]=analogRead(analogInPin);
delay(10); } for(int i=0;i<9;i++) { for(int j=i+1;j<10;j++)
{ if(buf[i]>buf[j]) { temp=buf[i]; buf[i]=buf[j];
}
 } } for(int i=2;i<8;i++) avgValue+=buf[i]; float
pHVol=(float)avgValue*5.0/1024/6; float phValue = -5.70 * pHVol +
21.34;

Serial.println(phValue);
Serial.print("pH");



 Serial.print(" C ");
 Serial.print(Celcius);

 Serial.print(voltage); Serial.print("V"); delay(10000);
}
```

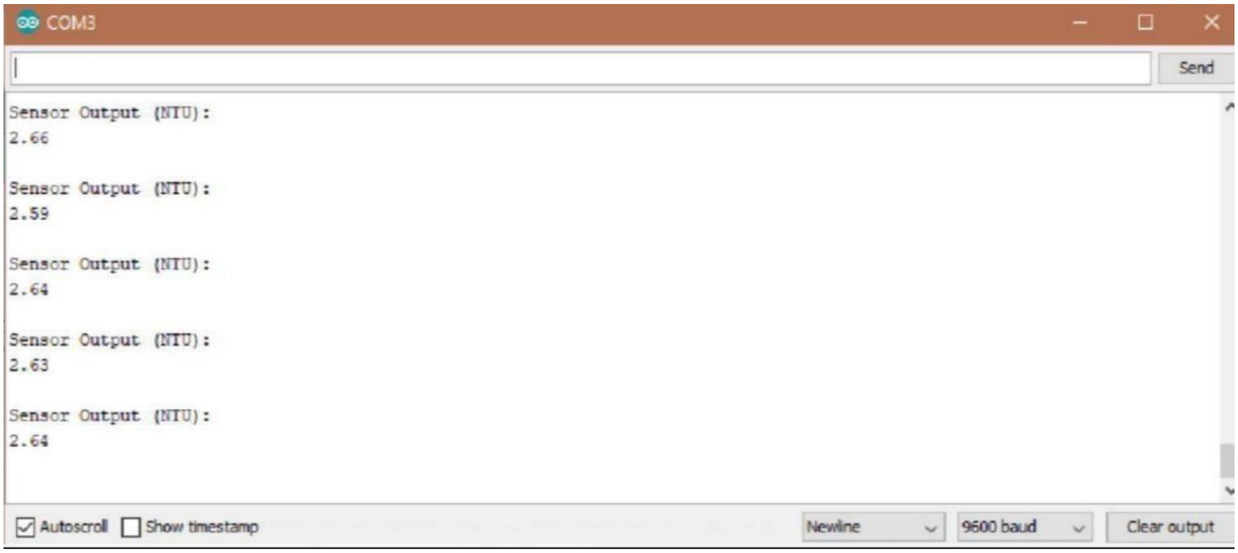
CODE IMPLEMENTATION:

```
import serial import time import csv import numpy as np import matplotlib.pyplot as plt ser =
serial.Serial('/COM6',9600) ser_bytes = ser.readline(10) print (ser_bytes) ser.flushInput() while
True:
 try:
 ser_bytes   =   ser.readline()     decoded_bytes  =   float(ser_bytes[0:len(ser_bytes)-
2].decode("utf-8")) print(decoded_bytes)

temp = float(decoded_bytes(1:3)) turb = float(decoded_bytes(4:6))
pH = float(decoded_bytes(6:8)) with open("test_data.csv","a") as f:
writer                    =                    csv.writer(f,delimiter=",")
writer.writerow([time.time(),decoded_bytes])                  except:
```

```
{ if(buf[i]>buf[j]) { temp=buf[i]; buf[i]=buf[j];
print("Keyboard Interrupt") ser.close() break() t = np.arange(0.0, 2.0,

0.01) s = 1 + np.sin(2*np.pi*t) plt.plot(t, s) plt.xlabel('time (s)')

plt.ylabel('Celsisus (C)')
plt.title('Temperature') plt.grid(True)  plt.savefig("Temperature.png")
Serial.begin(9600); sensors.begin(); int sensorValue = analogRead(A1);
voltage =sensorValue * (5.0 / 1024.0);
}
void loop(void)
{
 sensors.requestTemperatures();
 Celcius=sensors.getTempCByIndex(0);
                                        Fahrenheit=sensors.toFahrenheit(C
elcius); for(int i=0;i<10;i++)
{
 buf[i]=analogRead(analogInPin); delay(10);
}
for(int i=0;i<9;i++)
{
 for(int j=i+1;j<10;j++)
 {
 if(buf[i]>buf[j])
 {
 temp=buf[i]; buf[i]=buf[j]; buf[j]=temp;
 }
n = 256
X = np.linspace(-np.pi, np.pi, 256, endpoint=True) C,S = np.cos(X),
np.sin(X) plt.plot(X, C) plt.plot(X,S) plt.show()

print ("Visualization of real time sensor Data.") print("/n") while True:
 try:

 ser_bytes = ser.readline() decoded_bytes = float(ser_bytes[0:len(ser_bytes)-
2].decode("utf-8")) print(decoded_bytes) temp = float(decoded_bytes(1:3)) turb =
float(decoded_bytes(4:6)) pH = float(decoded_bytes(6:8)) with open("test_data.csv","a")
as f: writer = csv.writer(f,delimiter=",")
```

```
  np.arange(0.0, 2.0, 0.01) s = 1 + np.sin(2*np.pi*t) plt.plot(t, s)
plt.title('Temperature')  plt.grid(True)  plt.savefig("Temperature.png")
```

ARDUINO OUTPUT:



```
Sensor Output (NTU):
2.66

Sensor Output (NTU):
2.59

Sensor Output (NTU):
2.64

Sensor Output (NTU):
2.63

Sensor Output (NTU):
2.64
```

☑ Autoscroll □ Show timestamp Newline 9600 baud