REAL TIME WATER QUALITY MONITORING & CONTROL SYSTEM

DATA SET

ANALYSIS GIVES RANDOM VALUES OF PH, TURBIDITY & TEMPERATURE

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

#Provide your IBM Watson Device Credentials
organization = "shg55z"
deviceType = "abcd"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
```

```
        print ("led is on")
    elif status == "lightoff":
        print ("led is off")
    else :
        print ("please send proper command")


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()


# Connect and send a datapoint "hello" with value "world" into the cloud as
an event of type "greeting" 10 times
deviceCli.connect()


while True:
        #Get Sensor Data from DHT11
```

```
        temp=random.randint(90,110)
        turbidity=random.randint(60,100)
        ph=random.randint(6,10)

        data = { 'temp' : temp, 'turbidity': turbidity,'Ph':ph }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp,"ph=%s"%ph,
"turbidity = %s %%" % turbidity, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```