

SPRINT 2

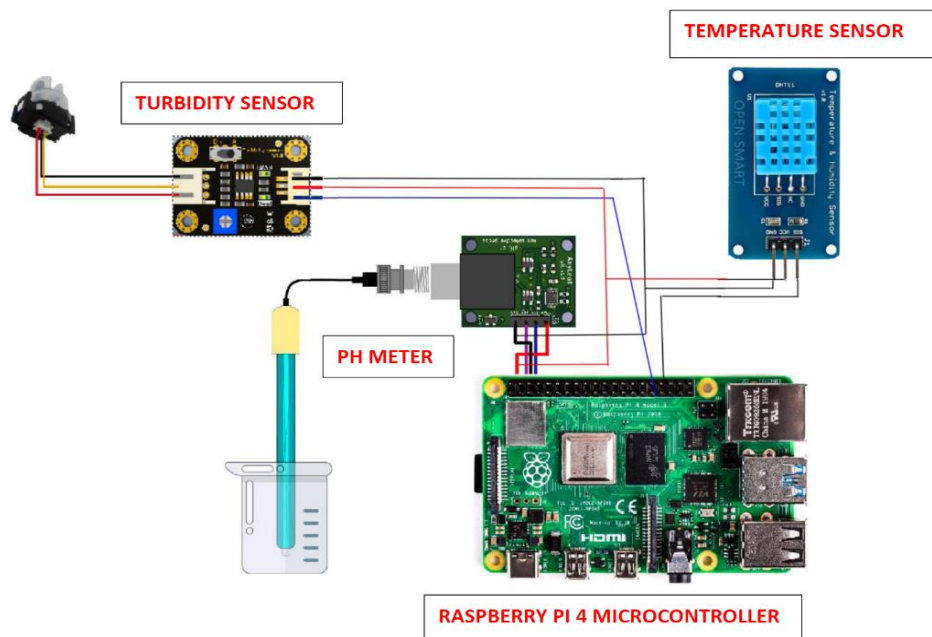
IBM CLOUD

The screenshot displays the IBM Watson IoT Platform interface. On the left, a code editor shows a Python script named `Test_Python_3.7.4.py` that generates random data for pH, turbidity, and temperature. The script is running on a Raspberry Pi, and the output is visible in the console window below the code editor. The console shows a series of log messages indicating that data is being published to the IoT platform.

On the right, the IBM Watson IoT Platform console shows a table of recent events. The table has two columns: **Event** and **Value**. The events are labeled `demo` and contain JSON-formatted data representing the sensor readings.

Event	Value
demo	{"pH":12,"turbid":93,"temp":87}
demo	{"pH":7,"turbid":873,"temp":94}
demo	{"pH":3,"turbid":204,"temp":19}
demo	{"pH":11,"turbid":304,"temp":77}
demo	{"pH":13,"turbid":16,"temp":50}

CIRCUIT DAIGRAM



CODING

```
Untitled - Notepad
File Edit View
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys
from twilio.rest import Client
import keys
client = Client(keys.account_sid, keys.auth_token)

organization = "lwkiac"
deviceType = "Microcontroller_Device_1"
deviceId = "00002"
authMethod = "token"
authToken = "sushi@123"

pH = random.randint(1, 14)
turbidity = random.randint(1, 1000)
temperature = random.randint(0, 100)

def myCommandCallback(cmd):
    print("Command Received: %s" % cmd.data['command'])
    print(cmd)

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(1, 14)
    turbidity = random.randint(1, 1000)
    temperature = random.randint(0, 100)

    data = {'pH': pH, 'turbid': turbidity, 'temp': temperature}
    def SMS():
        message = Client.messages.create(
            body="ALERT!! THE WATER QUALITY IS DEGRADED",
            from_=keys.twilio_number,
            to = keys.target_number)
        print(message.body)

    if temperature>70 or pH<6 or turbidity>500:
        SMS()

    def myOnPublishCallback():
        print("Published pH= %s" % pH, "Turbidity:%s" % turbidity, "Temperature:%s" % temperature)

    success = deviceCli.publishEvent("demo", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not Connected to ibmiot")
    time.sleep(5)
    deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
#Twilio Account Credentials
account_sid = 'Ac0e0b9bf43aa629b503bdd01d0962d465'
auth_token = '48c1a0ae0472038ab36d45e0d9fb6e7'
twilio_number = '+19804095xxx'
target_number = '+919940955xxx'
```

```
Untitled - Notepad
File Edit View
except Exception as e:
    print("caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()

while True:
    pH = random.randint(1, 14)
    turbidity = random.randint(1, 1000)
    temperature = random.randint(0, 100)

    data = {'pH': pH, 'turbid': turbidity, 'temp': temperature}
    def SMS():
        message = Client.messages.create(
            body="ALERT!! THE WATER QUALITY IS DEGRADED",
            from_=keys.twilio_number,
            to = keys.target_number)
        print(message.body)

    if temperature>70 or pH<6 or turbidity>500:
        SMS()

    def myOnPublishCallback():
        print("Published pH= %s" % pH, "Turbidity:%s" % turbidity, "Temperature:%s" % temperature)

    success = deviceCli.publishEvent("demo", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not Connected to ibmiot")
    time.sleep(5)
    deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
#Twilio Account Credentials
account_sid = 'Ac0e0b9bf43aa629b503bdd01d0962d465'
auth_token = '48c1a0ae0472038ab36d45e0d9fb6e7'
twilio_number = '+19804095xxx'
target_number = '+919940955xxx'
```

