# PROJECT DEVELOPMENT PHASE
# DELIVERY OF SPRINT PLAN_1(CODING)

| | |
|---|---|
| Date | 14 November 2022 |
| Team ID | PNT2022TMID07205 |
| Project Name | Project – SMART FARMER (IoT Enabled Smart Farming Application) |

**CODE:**
**publisheribm.py**

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

#Provide your IBM Watson Device Credentials
organization = "iotdevice1" # repalce it with organization ID
deviceType = "iotdevice1" #replace it with device type
deviceId = "qwerty123" #repalce with device id
authMethod = "token"
authToken = "qwerty123"#repalce with token

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:
      T=50;
      H=32;
        ot=45
      #Send Temperature & Humidity to IBM Watson
      data = {'d':{ 'Temperature' : T, 'Humidity': H,'objTemp':ot }}
      #print data
      def myOnPublishCallback():
          print (data, "to IBM Watson")

      success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
      if not success:
          print("Not connected to IoTF")
      time.sleep(1)

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

**pubsubibm.py**

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

#Provide your IBM Watson Device Credentials
organization = "iotdevice1" # repalce it with organization ID
deviceType = " iotdevice1" #replace it with device type
deviceId = "qwerty123" #repalce with device id
authMethod = "token"
authToken = " qwerty123"#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:
      T=50;
      H=32;
      #Send Temperature & Humidity to IBM Watson
      data = { 'Temperature' : T, 'Humidity': H }
      #print data
      def myOnPublishCallback():
         print ("Published Temperature = %s C" % T, "Humidity = %s %%" %
H, "to IBM Watson")

      success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
      if not success:
         print("Not connected to IoTF")
      time.sleep(1)

      deviceCli.commandCallback = myCommandCallback

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

**subscribeibm.py**

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device
```

```
#Provide your IBM Watson Device Credentials
organization = "iotdevice1" #replace the ORG ID
deviceType = " iotdevice1"#replace the Device type wi
deviceId = "qwerty123"#replace Device ID
authMethod = "token"
authToken = " qwerty123" #Replace the authtoken

def myCommandCallback(cmd): # function for Callback
	print("Command received: %s" % cmd.data)
	if cmd.data['command']=='lighton':
		print("LIGHT ON IS RECEIVED")

	elif cmd.data['command']=='lightoff':
		print("LIGHT OFF IS RECEIVED")

	if cmd.command == "setInterval":

		if 'interval' not in cmd.data:
			print("Error - command is missing required information:
'interval'")
		else:
			interval = cmd.data['interval']
	elif cmd.command == "print":
		if 'message' not in cmd.data:
			print("Error - command is missing required information:
'message'")
		else:
			output=cmd.data['message']
			print(output)

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:

        deviceCli.commandCallback = myCommandCallback

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